Privacy-Preserving Video Analytics

by

Francis Cangialosi

B.A., University of Maryland (2016)
B.S., University of Maryland (2016)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

June 2023

Authored by:	Francis Cangialosi Department of Electrical Engineering and Computer Science May 19, 2023
Certified by:	Hari Balakrishnan Fujitsu Professor of Electrical Engineering and Computer Science Thesis Supervisor
Accepted by:	Leslie A. Kolodziejski Professor of Electrical Engineering and Computer Science Chair, Department Committee on Graduate Students

Privacy-Preserving Video Analytics

by

Francis Cangialosi

Submitted to the Department of Electrical Engineering and Computer Science on May 19, 2023, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

As video cameras have become pervasive in public settings and accurate computer vision has become commonplace, there has been increasing interest in collecting and processing data from these cameras at scale ("video analytics"). While these trends enable many useful applications (such as monitoring the mobility patterns of cars and pedestrians to improve road safety), they also enable detailed surveillance of people at an unprecedented level. Prior solutions fail to practically resolve this tension between utility and privacy, as they rely on perfect detection of all private information in each video frame—an unrealistic assumption.

In this dissertation, we present Privid, a privacy-preserving video analytics system that aims to provide both a meaningful guarantee of privacy and an expressive, general query interface that is amenable to a wide range of analysts. In particular, Privid's privacy definition does not require perfect detection of private information, and its query interface allows analysts to provide their own arbitrary (untrusted) machine learning (ML) processing models.

The key takeaway from our evaluation is that Privid can provide a practical balance between privacy and utility: across a variety of queries over both real surveillance videos and a simulated city-wide camera network, Privid protects the appearance of all people with differential privacy, and maintains accuracy within 79-99% relative to a non-private system.

Thesis Supervisor: Hari Balakrishnan
Title: Fujitsu Professor of Electrical Engineering and Computer Science

Acknowledgments

It takes a village to produce a PhD thesis. Truly. I feel incredibly grateful and privileged for my village of mentors, colleagues, friends, and family who have shaped me, both during my PhD and all the years leading up to it. If not for them, none of this would have been possible.

I want to start by thanking my advisor, Hari Balakrishnan, for all of his mentorship and the many ways he supported me since I arrived at MIT seven years ago. I am especially grateful for Hari's patience and guidance as I meandered, working on different projects and trying to hone my interests throughout grad school. Hari continually pushed me to pursue my own interests rather than those of anyone else, even when that ultimately led me outside of his typical domain. Nevertheless, he always managed to provide sound guidance and critical insight into my work. Many of the lessons I learned from Hari were non-technical, subtle things that cannot be easily articulated on paper, but I am certain they made me a better researcher and person.

Ravi Netravali was also an advisor to me in every way except by title. Ravi and I first shared an office during his last year at MIT, and I feel very fortunate that we began chatting about research ideas just before he left. Little did I know that those chats would eventually lead to this thesis! Ravi collaborated with me closely on all aspects of this work and was extremely generous with his time and energy every step of the way. In particular, Ravi taught me a lot about refining and communicating technical ideas. His attention to detail and persistence on clarity were instrumental in making this work as polished as it is. On top of his technical contributions, Ravi was a boundless source of encouragement, positivity, wisdom, and life advice. I often went into conversations with Ravi feeling anxious about one thing or another, but always left feeling at ease.

I am grateful to Henry Corrigan-Gibbs for serving on my thesis committee and providing much-needed encouragement, reassurance, and advice near the end of my PhD. I am also grateful to Nickolai Zeldovich and Danny Weitzner for their helpful feedback and (constructive) criticism which kept me on the right track.

I am indebted to all of my collaborators on the Privid project. Venkat Arun was a consistent sounding board, always eager for a spontaneous chat, many of which strengthened the ideas in this thesis. In particular, Venkat helped to fill in my gaps in mathematical formalism and caught subtle mistakes in my proofs. On top of his technical contributions, I am thankful to have had Venkat as an officemate and close friend. I enjoyed all of our discussions–ranging from deeply technical to frivolous and everything in between–and the many jokes that kept our office environment light-hearted. Neil Agarwal played a key role in the implementation and evaluation of Privid. As anyone in systems knows, these things are subtly difficult and time-consuming. I especially appreciated Neil's enthusiasm and thoughtfulness; he always carefully considered the work we were doing, rather than blindly following suggestions. Junchen Jiang and Srinivas Narayana provided guidance and feedback from the very beginning, and were instrumental in crafting the problem and scope for Privid. I am

especially appreciative of the excitement, enthusiasm, and experience they brought to all of our project meetings. Anand Sarwate was always happy to chat with me on topics ranging from the philosophy of privacy to the technical weeds of a proof. Our conversations helped me to iron out bugs and gave me confidence in the theoretical foundations of Privid.

Before working on the contents of this thesis, I was fortunate to work with a number of great collaborators on a series of projects adjacent to congestion control: Deepti Raghavan, Akshay Narayan, Prateesh Goyal, Srinivas Narayana, Radhika Mittal, and Mohammad Alizdeh. Although these projects are not included in this thesis, they contributed to my growth and made me a better researcher, which in turn enabled this work. In particular, I grew as a software engineer and systems builder by working alongside Akshay and benefited tremendously from the many chats and words of encouragement from Prateesh.

My research was greatly enriched by the opportunity to collaborate with folks in industry:

I am grateful to Ranjeeth Dasineni for initiating my two-year collaboration with Facebook and the opportunity to bring our work on CCP [62] into the real world (i.e., Facebook's codebase). Udip Pant graciously supported this project, and provided not only technical advice, but also great conversations and friendship. I am also thankful for Yang Chi, Luca Niccolini, Matt Joras and all of the other members of the transport team who provided a great working environment and were always eager to help when I had a question.

During my last year, Hari provided me the opportunity to continue my work on privacy in a very practical setting with Cambridge Mobile Telematics. I greatly appreciate and admire his desire to prioritize doing the right work, above publication or financial motivations, and this collaboration was just one example of that. Every single person I interacted with at CMT was extremely welcoming, kind, and enthusiastic. In particular, I am grateful to Paresh Malalur for being both a mentor and core collaborator on my work at CMT. Paresh never hesitated to hop on a call outside of our scheduled meetings and was always ready dive into the weeds of an implementation detail or proof. I am especially appreciative of his patience with me whenever the work ventured into technical details that were outside of my area of expertise. Rather than taking the easier route of filling in gaps, he took the time to help me learn what was necessary so that I could fill them in myself.

I am extremely grateful to all of my colleagues from my days as an undergraduate at the University of Maryland for providing my foundations as a researcher. In particular, this starts with Dave Levin. When I expressed interest in thinking more about a class project during my sophomore year, Dave eagerly invited me to work with him. That project transitioned into a three-year collaboration, throughout the school years and summers, on three separate publications. From the very beginning, and before I deserved it, Dave gave me the attention and mentorship one would give a PhD student and also treated me as one of his peers. Dave taught me many invaluable lessons about how to both approach and execute good research. He also instilled in me a high bar for presentations and publications, and graciously taught me many of

his Keynote tricks. As a testament to his dedication, Dave helped me practice my first conference talk (IMC 2015 in Tokyo) more times than I can count and was apparently more nervous about the talk than I was myself. If not for Dave, I certainly would not have made it[1] to MIT. Furthermore, Dave gave me the opportunity to work with his fantastic group of collaborators outside of MIT, who were welcoming and gracious with their advice and encouragement: Alan Mislove, Christo Wilson, Dave Choffnes, and Bruce Maggs. I'm also grateful to Neil Spring and Bobby Bhattacharjee for their insightful and candid feedback on papers and presentations.

It was also in the UMD Systems lab that I met Matt Lentz, who has been both a close friend and "life advisor" ever since. Matt accompanied me to my first conference, and taught me, by example, the fundamentals of networking. Matt has consistently served as a role model and inspiration for me as a researcher, especially for his good taste and extreme attention to detail when it comes to figures and slides. He has sharpened my abilities through feedback on most of the visual content I've made. I am grateful for all of the feedback and advice he's given over the years, but most importantly, I am grateful for his friendship.

Thank you to my G982 officemates, NMS labmates, and G9 cohabitants, past and present (in alphabetaical order)—Venkat Arun, Arjun Balasingam, Inho Cho, Jon Gjengset, Prateesh Goyal, Pouya Hamadanian, Songtao He, Peter Iannucci, Pantea Karimi, Mehrdad Khani, Moein Khazraee, Sunghyun Kim, Derek Leung, Chenning Li, Alex Mallery, Hongzi Mao, Akshay Narayan, Srinivas Narayana, Arash Nasr-Esfahany, Vikram Nathan, Pari Negi, Ravi Netravali, Amy Ousterhout, Seo Jin Park, Deepti Raghavan Sudarsanan Rajasekaran, Harsha Sharma, Vibhaa Sivaraman, Will Sussman, Lily Tsai, Frank Wang, Lei Yang, and Zhizhen Zhong—for making the Stata center a fun and pleasant place to work, and for all of the conversations over the years. I have fond memories of our regular table tennis matches, badminton games, hikes, and trips to Tosci's. In particular, thank you to Vibhaa for all of her (often underappreciated) work organizing so many things for our group, including many of those social activities, and many thesis defenses. Thank you also for being a caring friend, listener, and sounding board for me through some difficult moments.

I am also very grateful for all of the administrative staff—Sheila, Angelly, Janet, and Alicia—who worked tirelessly behind the scenes to support all of the activities that fueled our research, including conference travel, funding, reimbursements, snacks, and group lunches.

Katie Lewis and David Palmer were my problem set partners and the reason I survived (and learned immensely from) David Karger's advanced algorithms course. We somehow fell into a tradition of going to A4 roughly once a semester and ordering the exact same two pizzas every single time. I am thankful that we somehow managed to keep that consistently through the rest of grad school and hope that there will be more in the future.

My community outside of MIT provided me with much-needed balance, perspective, and purpose.

[1]The jury is still out as to whether I am thanking or blaming him for this.

Symphony Church provided a warm and welcoming community for me in Boston from the moment I moved here. In particular, I want to acknowledge Pastor Barry, the worship team, the Cambridge family group, and the JP microchurch for their friendship, words of wisdom, and encouragement throughout all of the difficult seasons during my PhD. And thank you to Sarah Yoo for introducing me to Symphony in the first place.

My housemates in "3251"—Colin, Brian, and Su-Young—were my family in Boston, and living with them marked the moment when Boston began to feel like my new home. We prepared and shared countless meals together, watched dramas, played games, took trips, laughed a lot, and overall just did life together. I cherish all the memories we made and am indebted for all the emotional and moral support they provided throughout our time living together. I am thankful to Tony Lee for providing that home and unique opportunity for us.

Running has been foundational to my physical and mental health over the last few years of my PhD. Thank you to everyone who has shared the streets with me for a run at some point. In particular, I am grateful to my fellow Rat City Run Club members—James and Sunny—for inspiring my interest in marathons, and traveling across Europe with me to run the 2022 Berlin Marathon. And I am grateful for the 6:34am Volt group, KC, and the broader Heartbreak Running community for the many track workouts and long runs that improved my fitness and gave me energy for the rest of my workdays.

Finally, I am forever grateful to my entire family, including the extended Cangialosi family, for always believing in me and building me up, from the very beginning and continuously to this day. Thank you to my parents, for their unconditional love and endless desire to provide for me and help me in any way they can. Thank you to my in-laws, for welcoming me into their family with open arms and for breaking *all* of the negative Korean in-law stereotypes. And thank you to my beautiful wife, Su-Young, who has been my dearest friend, life partner, and number one cheerleader and advocate for the past 10 years, through all the ups and downs of my PhD. I am indebted to her for the many sacrifices she made to be by my side throughout this process. She often reminds me that less is more and that I shouldn't overthink or overcomplicate things. I am still working on that, but for everyone else's sake, I will condense all the acknowledgements she deserves into this brief sentiment: thank you for all the joy and happiness you've given me. There is truly never a dull moment with you! I feel confident in my career ahead knowing that I have you by my side.

Previously Published Material

The contents of this thesis extend a conference paper at NSDI 2022 [27], which was joint work with Neil Agarwal (Princeton), Venkat Arun (MIT), Junchen Jiang (University of Chicago), Srinivas Narayana (Rutgers), Anand Sarwate (Rutgers), and Ravi Netravali (Princeton).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

As technology has progressed, it has become easier and easier to collect data about our personal lives from more and more sources. Today, data can be collected from the usage of our laptop, the location and movement of our phone, the physical activity of our watch, the operation of our car, the usage of lights in our home, and from the security cameras that line our streets, to name just a few examples.

Data fuels the development of many of the important services we use, both directly (e.g., wellness apps analyze our physical movements to help us track our health and identify issue) and indirectly (e.g., the government collects traffic and mobility data to mediate traffic congestion). Collecting more of this data enables organizations to create better algorithms, better models, better results, and ultimately offer us better utility and new services.

However, this data covers an increasing fraction of our lives, both online and in the physical world. This means that our level of "privacy"—informally, the fraction of our lives that is *not* available to be scrutinized and judged—is quickly vanishing. In other words, there is an *inherent tension between utility and privacy*: the same data can power both new technology and better surveillance.

This makes it difficult when we have to decide whether or not to share data. Unfortunately, in most cases the options are black and white: trust the organization offering the service in exchange for data ("giving up" on privacy), or prioritize privacy by denying the service, "giving up" on the potential utility of it. But "trusting" an *entire* organization is precarious and difficult to reason about. There is always the chance that an organization could be outright malicious. But there are many more dangerous scenarios: they could hide malicious intent behind a genuine service, begin with good intent but succumb to mission creep [25], fail to implement their policies [36], sell data to untrusted parties [60], be subject to a data breach, or have some malicious employees internally.

However, the pressure to share data is great. There are often strong economic (an organization is willing to pay a lot for useful data) or moral (contributing data could help an organization on their quest to make the world a better place) incentives. It is easy to protect privacy by never sharing any data, but this is not desirable, and impedes a lot of exciting and useful technological progress.

Can we get the best of both worlds? In many of these cases, we just want to allow the organization to make some *inferences* about our data. Is there a way to enable organizations to make the inferences they need to carry out their services without giving them access to our data directly? In this dissertation, we explore that question for a particular emerging domain where the potential utility is high, but the threat of surveillance is particularly concerning: video analytics.

High-resolution video cameras are now pervasive in public settings [3, 5, 8, 1, 4, 75], with deployments throughout city streets, in our doctor's offices and schools, and in the places we shop, eat, or work. Traditionally, these cameras were monitored manually, if at all, and used for security purposes, such as providing evidence for a crime or locating a missing person. However, steady advances in computer vision [26, 53, 55, 77, 50] have made it possible to automate video-content analytics (both live and retrospective) at a massive scale across entire networks of cameras. While these trends enable a variety of important applications [2, 9, 11, 10] and have received much positive attention in the computer systems community [45, 23, 46, 54, 21, 41, 42, 39, 88], they also enable privacy intrusions at an unprecedented level [7, 76].

As a concrete example, consider the operator for a network of city-owned cameras. Different organizations (i.e., "analysts") want access to the camera feeds for a range of needs: (1) health officials want to measure the fraction of people wearing masks and following COVID-19 social distancing orders [37], (2) the transportation department wants to monitor the density and flow of vehicles, bikes, and pedestrians to determine where to add sidewalks and bike lanes [18], and (3) businesses are willing to pay to analyze shopping behaviors to improve their operations [16].

Unfortunately, freely sharing the video with these parties may enable them to violate the privacy of individuals in the scene by tracking where they are, and when. For example, the "local business" may actually be a bank or insurance company that wants to track individuals' private lives for their risk models, while well-known companies or government agencies may succumb to mission creep [15, 14, 17]. Further, any organizations with good intentions could have employees with malicious intent who wish to spy on a friend or co-worker [12, 13]. In other words, the camera owner's options are at opposite ends of the spectrum and are both undesirable: freely sharing the video gives up on privacy, while not sharing it gives up on utility.

In this dissertation, we offer a third option that strikes a practical balance between privacy and utility: a new video analytics system that allows analysts to process video data for a wide range of use cases, while providing a privacy guarantee ensuring citizens cannot be tracked. As a result, we show that it is indeed possible to build a general video analytics system that serves both analysts *and* citizens. Before we describe the gap we seek to fill in prior work (§1.3), we define our problem and goals more precisely to put everything into context.

1.2 Problem Definition and Goals

Video analytics broadly refers to the application of computer vision algorithms to large amounts of video data to extract insights about its contents. A typical video analytics setting involves four parties (though some may pertain to the same entity):

- A **Video Owner**, who operates one or more cameras and owns the video data they capture,
- **Individuals**, whose behavior and activity are observed by these camera(s),
- **Analyst**(s), who wish to analyze the video data, and a
- **Compute Provider**, who executes the analyst's computation.

Analysts are interested in questions about the video data, such as "How many people entered store X each hour?" or "Which roads suffered from the most accidents in 2020?" (see Chapter 5 for more specific examples). Analysts express a question as a "query", which is a function that takes a video as input and outputs the answer to the question (e.g., a statistic)[1]. A query typically involves a pipeline of separate algorithms—some using machine learning, and others using traditional programming logic—where the output of one provides the input for the next. For example, to answer the question "what is the average speed of red cars traveling along road Y?",

[1]Related work typically defines a query as returning "intermediate" results (e.g., bounding boxes). Our definition is distinct in that we consider a query to return the final answer to the high-level question at hand, and to encompass *all* of the necessary computation.

the "query" would include an object detection algorithm to recognize cars, an object tracking algorithm to group them into trajectories, an algorithm for computing speed from a trajectory, and logic to filter only the red cars and average their speeds.

In this work, we are concerned with the dilemma of a Video Owner: they would like to enable a variety of (untrusted) analysts to run queries over their videos, as long as the *results* do not infringe on the privacy of any individuals who appear in the videos. Informally, privacy "leakage" occurs when an analyst can learn something about a specific individual that they did not know before executing a query. To practically achieve these properties, a system must meet three concrete goals:

Goal 1: Formal notion of privacy. The system's privacy policies should formally (rigorously) describe both the type and amount of privacy that could be lost through a query. Given a privacy policy, the system should be able to provide a *guarantee* that it will be enforced, regardless of properties of the data or query implementation. Without such a guarantee, it is difficult for anyone to reason about what privacy is actually being provided and whether or not it is sufficient.

Goal 2: Maximize utility for analysts. The system should support (i.e. with reasonable accuracy) any query whose final *result* does not infringe on the privacy of any individuals. A key factor of the potential for video analytics is the diversity of information it captures and thus the range of applications it could enable. A system that achieves high accuracy and privacy for a very limited set of queries would leave the vast majority of use cases unaddressed and thus would not ease the tension between privacy and utility in practice. Further, if accuracy loss is introduced to improve privacy for a given query, it should be possible to *bound* that loss (relative to running the same query over the original video, without any privacy preserving mechanisms). Without such a bound, analysts would be unable to rely on any provided results and would be unlikely to use the system.

Goal 3: "Bring Your Own Model". Computer vision models are at the heart of modern video processing. However, there is not one or even a discrete set of models for all tasks and videos. Even the same task may require different models, parameters, or post-processing steps when applied to different videos. In many cases, analysts will want to use models that they trained or fine-tuned themselves, especially when they have proprietary data for their task. Thus, in order for a system to be practical, it must allow analysts to use their own video-processing models.

Scope. The class of analytics queries we seek to enable is distinct from *security-oriented* queries (e.g., finding a stolen car or missing child), which *require* identification of a particular individual, and are thus directly at odds with individual privacy. In contrast, analytics queries involve searching for patterns and trends in large amounts of data; intermediate steps may operate over the data of specific individuals, but they do not need to distinguish individuals in their final aggregated result.

1.3 Prior Solutions

Over many years, the computer vision and systems communities have proposed a wide variety of approaches to address privacy concerns for visual data. However, most use some variant of the same strategy: first find *all* private information in the video, then hide it. Unfortunately both steps fundamentally conflict with the goals laid out in the previous section.

The first step alone can be unrealistic in practice (§2.1); it requires: (1) an explicit specification of all private information that could be used to identify an individual, and then (2) the ability to spatially *locate* all of that information in *every* frame of the video. For the former, while some information is obviously identifying (e.g., one's face), it is difficult to enumerate the *full* set of objects that could identify an individual in all scenarios (e.g., a distinctive backpack or bike, the individual's walking gait, their car, or even the building they exit). For the latter, even state-of-the-art CV algorithms are imperfect [6], especially in the challenging conditions (i.e., poor lighting, distant objects, frequent occlusions) that are prevalent in public video.

Further, if these approaches cannot find some private information, they fundamentally cannot *know* that they missed it. Taken together, they can provide, at best, a conditional and brittle privacy guarantee such as the following: "if an individual is only identifiable by their face, and their face is detectable in every frame of the video by the system's specific CV model implementation in the specific conditions of this video, *then* their privacy will be protected." In other words, these approaches fail to provide a meaningful guarantee of privacy for individuals, because whether or not any given individual will be protected depends heavily on the data and implementation. CV is rapidly advancing, and there is no systematic way to determine which information is or is not detectable in a video. Just because the system's CV model cannot detect a particular individual does not mean a malicious analyst's model would not be able to do so.

The second step precludes many common applications. Often the objects whose individual privacy we want to protect (people, cars, etc.) are the very objects that we wish to analyze in aggregate at the population level. By obscuring the individuals, we prevent the analysis of them in aggregate. A careful balance here is crucial; even if a system significantly reduces privacy threats, if it is not practical for *analysts* it will never be chosen over the black and white options of all-or-nothing privacy, and thus will not ease any privacy-utility tensions in practice.

On the other hand, Differential Privacy (DP) [34] has arisen as the gold standard framework for devising rigorous and meaningful privacy guarantees in many settings. Although DP has been studied extensively and applied to many adjacent problems, the framework does not directly apply to the *video analytics setting.* In simple terms, creating a differentially-private version of an algorithm requires analytically bounding the "sensitivity" of the algorithm, which is the amount any single individual in the input could contribute to the computation's output. This is incongruent with video analytics data for two reasons: (1) videos do not directly encode the individuals they observe (they can only be inferred from pixels using a computer vision algorithm), and (2) modern video processing relies on custom machine learning models (§1.2), but it is impractical to analytically bound all the ways an individual could impact a single model, much less the full set of models an analyst may want to use.

1.4 Contributions

The primary goal of this thesis is to bring the rigorous theory of differential privacy to the video analytics setting in practice. Towards this end, we make the following contributions:

- We design a new variant of differential privacy tailored to the video analytics setting: *event-duration privacy.* The key idea with this definition is that we take an entirely different approach to avoid the issues that plague prior approaches: rather than defining privacy over people, we define it over durations of time, which we use as a proxy for the existence of people. Importantly, this definition has a clear interpretation, well-defined semantics, and it can be enforced without knowledge of the latent private information in the video.

- We design a system, Privid, that provably satisfies event-duration privacy while supporting the use of arbitrary analyst-supplied (and thus untrusted) ML models. Privid provides a custom SQL-like interface that is familiar to analysts and can express a wide range of queries. We believe these properties will minimize the barrier to entry for analysts, and will make the key ideas in Privid amenable to inclusion in future video analytics systems.

- We implement Privid and evaluate it using a variety of queries across different objects and videos. The key takeaway from our evaluation is that, even when parameterized to protect the appearance of *all* individuals in standard surveillance videos, Privid can achieve accuracy close (within 1-21%) to that of a non-private system. We believe these results provide evidence that differetially-private guarantees could be feasible in the video analytics setting.
- We discuss a variety of concerns that arise when using Privid in practice, from both the video owner and analyst perspective, including parameter tradeoffs and best practices for designing queries to maximize utility.

1.5 Key Takeaways

There are two key takeaways from this dissertation:

Rigorous privacy guarantees can exist in practice. Often times, rigorous privacy guarantees, such as those based on differential privacy, are seen as difficult to deploy in practice, as they impose restrictions on queries, and degrade utility. In fact, sometimes the only way to satisfy such a guarantee is to entirely impede utility. This work offers some hope that, even in scenarios where it does not seem obviously achievable at first, such as video analytics, rigorous privacy guarantees can be satisfied without giving up on utility. In our case, the key to making progress was re-defining the query interface.

It's all about the interface. It is easy to take interfaces as a given. In typical video analytics systems, the "interface" is that queries output a primitive, such as bounding boxes, that can then be used to fuel downstream tasks. In typical visual privacy works, the "interface" is just the video itself. They assume computations need direct access to a video, and thus they aim to make a "safe" (denatured) version. Both of these interfaces evade a formal privacy guarantee. With Privid, we make progress by reframing our view of the interface: ultimately, analysts often do not care about bounding boxes directly, just the output of some computation over those bounding boxes. By creating an interface that requires analysts to specify their entire end-to-end query, we create a balance between query interpretability (for the purpose of managing privacy) and query expressivity (for the purpose of maximizing versatility for analysts).

Chapter 2

Prior Work

In this chapter, we review prior work on related privacy mechanisms to motivate our claim that a new approach is needed for the video analytics scenario.

2.1 Visual Denaturing

The predominant approach to privacy preservation with video data is *denaturing* [67], whereby systems aim to obscure (or, "denature") any private information in the video before releasing it for analysis. In principle, if nothing private is left in the video, then privacy concerns are eliminated. To the best of our knowledge, all prior work uses one or more [87] of the following strategies:

1. "Masking": these approaches [81] simply replace the bounding box (or silhouette) of an object with black (or some other fixed color) pixels. The aim is to entirely remove the information in the bounding box.

2. "Masking with metadata": these approaches also remove the pixels of a bounding box, but they attempt to replace it with some useful bits of information. Some replace it with metadata about the object that was masked, others [28, 84] attempt to replace it with encrypted bits of the object. This would allow authorized parties to recover the masked portions.

3. "Blurring" (or more generally, resolution reduction): these approaches blur either the pixels inside a bounding box (e.g., I-pic [20]) or the entire frame (e.g., Dai et. al [30]) rather than removing anything entirely. The intention is to retain some of the information, while ensuring it cannot be identified. However,

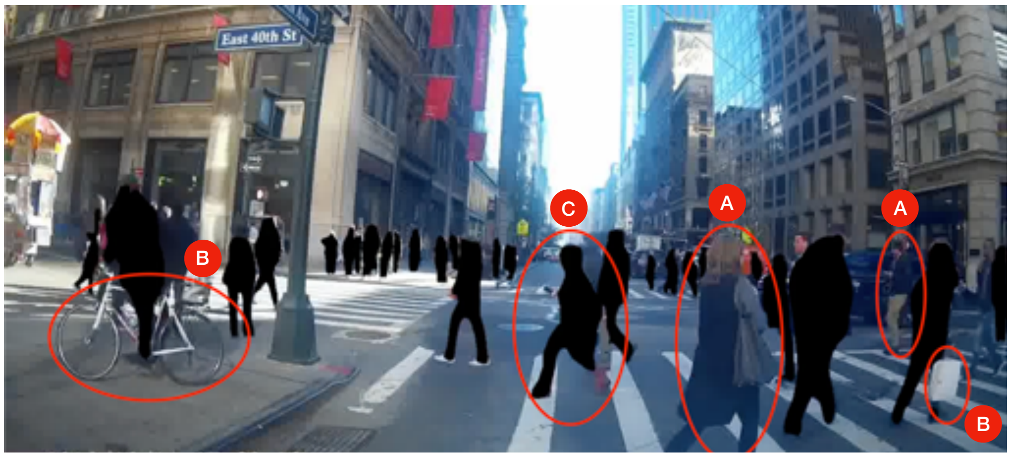


Figure 2-1: A video clip after "masking" denaturing (using silhouettes) exemplifying some of its shortcomings: (A) entirely missed detections, (B) potentially-identifying objects not incorporated in privacy definition (e.g., bag or bike), (C) silhouette may reveal walking gait, which could be uniquely identifying [65, 61].

prior work has shown that blurring is reversable [64, 52, 63] and low resolution can be upsampled using super-resolution techniques [47], which defeats its purpose.

4. "*k*-same": these approaches [64] aim to ensure each face or object is indistinguishable from at least k other objects, e.g. by replacing each of the objects with an average of the pixels of those k objects.

5. "Inpainting": these approaches [29] aim to "remove" an object by replacing a bounding box with the most likely "background", i.e. what would have been observed if the object did not exist at all.

6. "Adversarial perturbations": these approaches [74, 71] aim to add noise to the pixels of an image that are imperceptible to humans, but sufficient to fool an ML model and prevent it from recognizing objects.

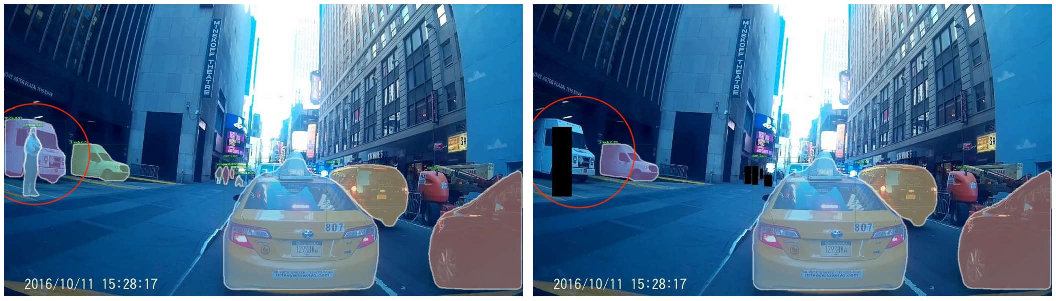
Unfortunately, all of these denaturing approaches share the same fundamental issue: they require *perfectly* accurate and comprehensive knowledge of the spatial locations of private information in *every frame* of a video. Any private object that goes undetected, even in just a single frame, will not be obscured and thus directly leads to a leakage of private information.

To detect private information, one must first semantically define *what* is private, i.e., what is the full set of information linked, directly or indirectly, to the privacy of each individual? While some information is obviously linked (e.g., an individual's face), it is difficult to determine *all* such information for all individuals in all scenarios. For instance, a malicious analyst may have prior information that a Video Owner does not, such as knowledge that a particular individual carries a specific bag or rides a unique bike (e.g., Figure 2-1-B). Further, even with a semantic definition, detecting private information is difficult. State-of-the-art computer vision algorithms commonly miss objects or produce erroneous classification labels in favorable video conditions [89]; performance steeply degrades in more challenging conditions such as poor lighting, distant objects, and low resolution, all of which are common in public video. These techniques give a false sense of privacy. While they certainly "appear" to make objects unrecognizable, it does not mean that a CV model could not be trained to recognize the object. Taken together, the problem is that denaturing systems cannot guarantee whether or not a private object was left in the video, and thus fail to provide a formal notion of privacy (violating Goal 1).

Denaturing also falls short from the analyst's perspective. First, it inherently precludes (safe) queries that aggregate over private information (violating Goal 2). For example, an urban planner may wish to count the number of people that walk in front of camera A and then camera B. Doing so requires identifying and cross-referencing individuals between the cameras (which is not possible if they have been denatured), but the aggregate count may be large and safe to release.[1] Second, denatured objects are not naturally occurring and thus video processing pipelines are not designed to handle them. If the analyst's processing code and models have not been trained explicitly on the type of denaturing the Video Owner is employing, it may behave in unpredictable and unbounded ways (violating Goal 2). For example, in Figure 2-2, we show an example where masking people prevented the detection of nearby objects, and in Figure 2-3, we show examples where masking people allowed some objects to be discovered that were originally missed. Thus, even in cases where denatured objects do not directly conflict with the query at hand, analysts would need to *re-train* their CV models with denatured objects in the training set (conflicting with Goal 3). Given the immense resources required to train accurate vision models, this requirement would be a huge barrier in real deployments.

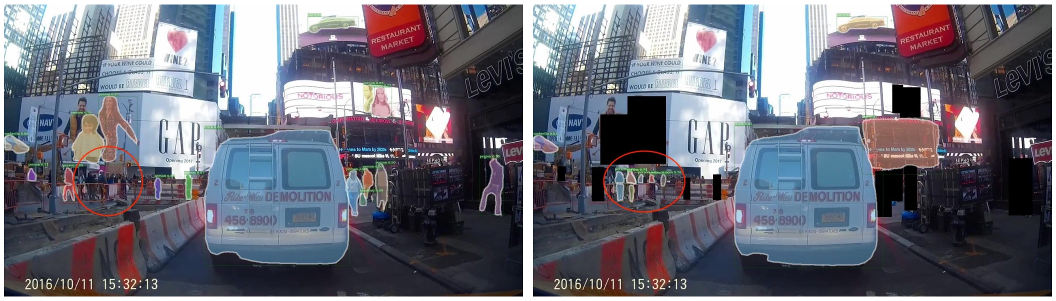
A few recent proposals (e.g. Wu et. al [86]) aim to address the utility shortcomings of denaturing by training a model to learn an optimal denaturing transform. The primary limitation of such approaches is that they require explicitly specifying a set of target tasks up front when training the denaturing method. This means they would

[1]As a workaround, the Video Owner could annotate denatured objects with query-specific information, but this would conflict with Goal 3.

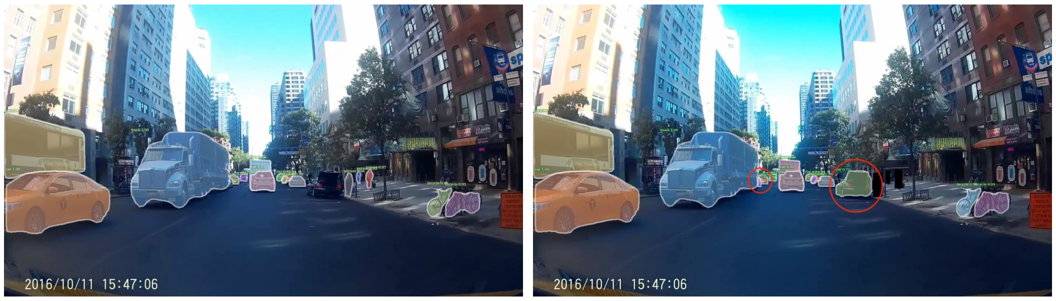


(a) Frame 3: Before masking (b) Frame 3: After masking

Figure 2-2: The left column shows the output of the Detectron2 [38] object detection model, where detected objects (people and cars) are covered with a colored silhouette. Example where masking some objects impedes the detection of other nearby non-masked objects. The car in the red circle is not detected after the intersecting person is masked.



(a) Frame 1: Before masking (b) Frame 1: After masking



(c) Frame 2: Before masking (d) Frame 2: After masking

Figure 2-3: For each row, the left column shows the output of the Detectron2 [38] object detection model, where detected objects (people and cars) are covered with a colored silhouette. For the right column, we replace the pixels of all detected objects with a black box, and show the output of the detection algorithm on this modified frame. The red circle in each figure highlights objects that were visible both before and after denaturing, but were only *detected* by the CV algorithm after denaturing (compare left and right for the same row).

not support new tasks (Goal 2) or even new models for the same task (Goal 3).

Synthetic Videos. One tempting approach used in other domains is the idea of generating and releasing a synthetic dataset (or in this case, video stream) rather than the real one. Ideally, if this video captures key properties of the original video data, it can still be used by analysts, and if it does not contain any actual people, then there are no privacy concerns and it can be released safely. Unfortunately, there are a number of fundamental issues with this approach. First, these approaches require modeling invariants of the data. A key characteristic of video data is that it captures a wide variety of information and complex interactions between objects. While it may be possible to model (and thus expose to an analyst) a few invariants, it would not be feasible to model all possible invariants, and anything not modeled could not be reliably queried by an analyst (Goal 2). Second, generating synthetic data is computationally expensive. Another key characteristic of video analytics is scale and liveness: there are many cameras, each camera is recording 24/7, and many are high resolution. Although not a technical limitation, in practice it would be infeasible to generate synthetic video proportional to the amount of real video data being recorded.

2.2 Systems

Denaturing Systems. In the previous section, we reviewed a variety of denaturing mechanisms for providing privacy, primarily developed in the computer vision community. In the systems community, many of the proposals (e.g., Wong et. al [83] and PECAM [84]) aim to address the challenges with *operationalizing* one of these denaturing techniques. These papers address a variety of related issues, but when it comes to protecting the privacy of citizens from analysts, they use denaturing, and thus all of the issues related to both privacy and utility from the previous section apply here.

Edge Computing. A general systems (rather than visual) approach suggested by a variety of different systems (e.g. EdgeEye [56]) is to move video processing to the "edge." While the edge can refer to many different things in different contexts, it typically means processing the video either directly on the camera itself, or at a physically-proximate server (e.g., for a traffic camera, the server may be at the traffic pole). While this vaguely "reduces" potential privacy issues or makes compromising privacy "more difficult," it does not prevent privacy issues or guarantee protection (violating Goal 1): it is still possible to deploy surveillance algorithms at the edge and send back the results! For example, the deployed model could detect faces and send back a stream of (name, location, activity). Nothing at the technology layer moderates this output, only policy. Even if the recipient of the data does not get to see the video directly (which is the only stated goal of the edge paradigm), arbitrary control over the computation could allow them to compromise citizens' privacy.

Pre-Defined Queries. Some real deployments of cameras for video analytics sidestep the privacy issues we target by combining edge computing with a pre-defined (and agreed upon) list of aggregate results. The results are computed at the edge, and only the results are released. One example is the City of Boston's 2016 Vision Zero pilot project, where Verizon installed (and managed) 6 cameras at traffic intersections. A public notice [78] states that the cameras process all video at a server mounted on the same light pole, and only aggregate counts of traffic statistics are sent over the network to city data analysts. The benefit of this approach is that it does not require any noise in order to protect privacy. The downside is that it is not a general system (violating Goals 2 and 3). The City of Boston must rely on Verizon to properly implement any computer vision tasks. Allowing city analysts to supply their own models for the pre-defined tasks (Goal 3), or more generally allowing analysts to come up with new queries (Goal 2) could not be supported without blindly trusting the analysts. This is exactly the issue Privid aims to solve.

Privacy in Cloud Environments. Visor [70] addresses privacy concerns that arise when video analytics are executed in cloud environments. In their threat model, the Video Owner trusts the analyst, but does *not* trust the compute provider. They assume both the Video Owner and analyst have full access to the video, but want to preserve the confidentiality of both the Video Owner's video and the analyst's ML model from cloud co-tenants and the cloud provider themselves. These concerns are orthogonal to our scenario, where we assume the cloud provider is trusted, but the analyst is not. Further, their techniques are complimentary to Privid: a practical deployment of Privid in the cloud could use Visor to run the analyst's ML models.

In summary, all prior approaches from the vision and systems communities fail to provide a *balance* between privacy, utility, and generality. Focusing on privacy at the cost of the other two does not actually solve the problem at hand: if analysts are unwilling to use a privacy-preserving system, then we are back at the same scenario where data is simply not shared and video data cannot be leveraged to its full potential. As stated in our goals (§1.2), we believe that the way forward for privacy-preserving video analytics involves not just privacy mechanisms, but an equal focus on utility and generality. Thus, we seek a general system that enables analysts to use their own code and models, and to process the real unaltered video data, while still providing privacy. To achieve this, we turn to the theory of differential privacy.

2.3 Differential Privacy

Differential Privacy (DP) is a formal definition of privacy for traditional databases [33]. It enables analysts to compute aggregate statistics over a database, while protecting the presence of any individual entry in the database. DP is not a privacy-preserving mechanism itself, but rather a goal that an algorithm can aim to satisfy. Informally speaking, an algorithm satisfies DP if adding or removing an individual from the input database does not noticeably change the output of computation, almost as if any given individual were not present in the first place. More precisely,

Definition 2.3.1. Two databases D and D' are *neighboring* if they differ in the data of only a single user (typically, a single row in a table).

Definition 2.3.2. A randomized algorithm $\mathcal{A}$ is *ϵ-differentially private* if, for all pairs of neighboring databases (D, D') and all $S \subseteq \mathit{Range}(\mathcal{A})$:

$$\Pr[\mathcal{A}(D) \in S] \leq e^{\epsilon}\Pr[\mathcal{A}(D') \in S] \tag{2.1}$$

A non-private computation (e.g., computing a sum of bank balances) is typically made differentially private by adding random noise sampled from a Laplace distribution to the final result of the computation [33]. The scale of noise is set proportional to the *sensitivity* (Δ) of the computation, or the maximum amount by which the computation's output could change due to the presence/absence of any one individual. For instance, suppose a database contains a value $v_i \in V$ for each user i, where $l \leq v_i \leq u$. If a query seeks to sum all values in V, any one individual's v_i can influence that sum by at most $\Delta = u - l$, and thus adding noise with scale $u - l$ would satisfy DP.

Challenges. Determining the sensitivity of a computation is the key ingredient of satisfying DP. It requires understanding (a) how individuals are delineated in the data, and (b) how the aggregation incorporates information about each individual. In the tabular data structures that DP was designed for, these are straightforward. Each row (or a set of rows sharing a unique key) typically represents one individual, and queries are expressed in relational algebra, which describes exactly how it aggregates over these rows. However, these answers do not translate to video data; we next discuss the challenges in the context of several applications of DP to video analytics.

Regarding *requirement (a)*, as described in §2.1, it is difficult and error-prone to determine the full set of pixels in a video that correspond to each user (including all potentially identifying objects).

Regarding *requirement (b)*, typical video processing algorithms (e.g., ML-based CV models) are not transparent about how they incorporate private objects into their results. Thus, without a specific query interface, the "tightest" possible bound on the sensitivity of an arbitrary computation over a video is simply the entire range of the output space. In this case, satisfying DP would add noise greater than or equal to any possible output, precluding any utility (violating Goal 2).

Given that DP is well-understood for tables, a natural idea would be for the Video Owner to use their own (trusted) model to first convert the video into a table (e.g., of objects in the video), then provide a DP interface over *that table*[2] (instead of directly over the video itself). However, in order to provide a guarantee of privacy, the Video Owner would need to completely trust the model that creates the table. This entirely precludes using a model created by the *untrusted* analyst (violating Goal 3).

We now review a few lines of work that are adjacent, but do not provide the link between DP and video analytics.

Synthetic Videos. Two prior approaches, Verro [79] and VideoDP [80], use DP to guide the creation of a synthetic video. VideoDP [80] aims to generate a video whose pixels do not depend too much on any private object. Verro [79] first removes all objects from the scene using inpainting techniques then inserts avatars to replace the true objects, ensuring that no individual influences the set of avatars too much. However, both approaches define neighboring databases over the set of objects detected by a computer vision algorithm ([80]-§2.1, [79]-§2.2). As a result, they suffer from both of the shortcomings of denaturing approaches described above: their privacy definition is linked to the success of a computer vision algorithm, and they remove information about the target objects that are necessary for many queries.

DP for Streaming Data. Video is a form of streaming data. While there exist a wide variety of approaches for providing DP over continuous streams of time-series data, to the best of our knowledge, all assume the private individuals are explicitly delineated in the data, which is not true for video. The closest definition to our own is from Kellaris et. al [48], which assumes the database is an infinite stream of binary

[2]This would be equivalent to adding DP to an existing video analytics interface, such as [45, 23], which treat the video as a table of objects.

matrices D_t of pre-defined users and events for each timestep t. It then introduces the notion of w-event privacy over this infinite stream, which protects the presence of a semantic event that occurs within a series of w consecutive timesteps. For example, if a user execute a series of events that form a trajectory in a location service, this definition would protect any trajectories of length less than w. While this is similar to our notion, it does not have any semantic meaning in the context of video due to the reliance on pre-defined individuals. In particular, how should we choose w for video? What is a bound on the portion of a video that any one individual could occupy? Our notion of event-duration privacy can be viewed as an answer to this question. Further, they do not support untrusted user-defined functions, which are a key requirement (Goal 3) and contribution of Privid.

DP for Machine Learning. Papers at the intersection of privacy and machine learning typically focus on model *training*. One body of work broadly focuses on ensuring the machine learning models do not retain too much information about the training data, and thus cannot identify individuals in the training data during inference. Another body of work, typically referred to as *federated learning* takes this one step further and aims to ensure that the training data is never centralized in the first place by training the model in a secure distributed fashion. However their end goal is the same, to protect the privacy of the individuals in the training data. The common thread among both pieces of work is that their focus is on the model itself. In this thesis, we are not concerned with the contents of the model, but rather *how it is used*; we are concerned with the privacy of the production data at the inference (or runtime) phase, rather than the training phase. As stated in Goal 3, we are given a model which has already been trained by the analyst, so training-based techniques do not apply here. Even if a model was trained in a privacy-preserving fashion that protects it from being used to reverse-engineer the training data, it could still be used to identify individuals at inference time and thus contribute to privacy leakage in our setting.

Subsample and Aggregate. The technique of splitting data into disjoint subsets and re-aggregating it to provide privacy is not new. Dwork and Roth [35] (Chapter 7.1) introduce the idea of "Subsample and Aggregate," where a standard database x is randomly partitioned into m blocks B_i, each block is processed by some function f, and then the $f(B_i)$ are aggregated using a standard differentially-private mechanism. The Airavat system [73] broadly leverages this idea to create a privacy-preserving version of MapReduce [31]. In both cases, the target dataset is a traditional database, and thus they cannot be readily applied to video data. In particular, they do not address the key questions of how video should be divided, how privacy should be defined, and how to compute a bound on the sensitivity of a video. Our contribution is to provide answers to these questions and build a general-purpose video analytics system that puts them all together.

Chapter 3

A New Privacy Definition: Event-Duration Privacy

In this chapter, we introduce “event-duration privacy,” a new definition of privacy motivated by the video analytics setting. We begin with the intuition behind the definition (§3.1), then state it formally (§3.2). We discuss why the definition is practical (§3.3), how to interpret the guarantees it provides (§3.4), and finally *why* the definition is formulated exactly as it is (§3.5).

3.1 Intuition

Differential privacy (DP) aims to enable aggregate computations over an entire population that do not reveal information about any one of the individuals that comprise that population. Creating a new privacy definition for a new scenario within the framework of DP requires one to: (a) define the structure of the database (population), then (b) define a notion of “neighboring databases,” i.e. what it means for two databases to differ by at most one “individual” of the population.

The key challenge when applying DP to video analytics is the fact that video does not directly encode the private individuals it contains, and thus we cannot define neighboring databases as usual. We can first use an (imperfect) CV algorithm to locate private individuals, but if we use this as the basis for our definition, then our guarantee will be contingent on the success of the CV algorithm.

To remove this dependence, we propose an alternative definition of privacy. Our key insight is based on two observations: (1) a large body of video analytics queries involve processing the data of individual people and objects, but they ultimately aggregate all of this data into a statistic, and (2) they typically aggregate over durations of video (e.g., hours or days) that far exceed the duration of any one individual in the scene (e.g., seconds or minutes) [45]. Thus, we reframe the approach to privacy to instead focus on the temporal aspect of private information in video data, i.e., *how long* something is visible to a camera: the video is the database, time units are "individuals," and we define neighboring databases based on durations of time as a *proxy* for the existence of people.

This notion of privacy has three benefits. First, it decouples the definition of privacy from its enforcement. Video naturally encodes units of time, and thus the enforcement mechanism does not need to make any decisions about what is private or find private information to protect it; everything (private or not) captured by a duration bound can be protected. Second, a duration bound that captures a set of individuals implicitly captures and thus protects any information visible for the same (or less) time *without specifying it* (e.g., an individual's backpack, or even their gait). Third, protecting all individuals in a video scene requires only specifying their *maximum* duration, and estimating this value is far more robust to the imperfections of CV algorithms than precisely locating those individuals and their associated objects in each frame. For example, even if a CV algorithm misses individuals in some frames (or entirely), it can still capture a representative sample and piece together trajectories well enough to estimate their duration (§3.3).

3.2 Formal Definition

We consider a video V to be an arbitrarily long contiguous sequence of frames, sampled at a fixed f frames per second, and recorded directly from a camera (i.e., unedited). A "segment" $v \subset V$ of video is a contiguous subsequence of those frames. The "duration" of a segment $d(v)$ is measured in real time (seconds), as opposed to frames. An "event" e is abstractly *anything* that is visible within the camera's field of view. A set of video segments *fully contain* an event if the event is not visible in any frames outside of those segments.

As a running example, consider a video segment v in which individual x is visible for 30 seconds before they enter a building, and then another 10 seconds when they leave some time later. The "event" e of x's visit (or, their "presence") is comprised of one 30-second segment (e_{enter}), and another 10-second segment (e_{leave}).

Definition 3.2.1 ((ρ, K)-bounded events)**.** An event e is (ρ, K)-bounded if there exists a set of $\leq K$ video segments that fully contain the event, and each of these segments individually have duration $\leq \rho$.

Definition 3.2.2 ((ρ, K) bound tightness)**.** Consider an event e that is (ρ, K)-bounded. If e is also (ρ', K')-bounded for some $\rho' \leq \rho$ and $K' \leq K$, then the (ρ', K') bound is "tighter". Any bound of $(\rho'' \geq \rho, K'' \geq K)$ is "looser". If e is exactly (ρ, K)-bounded (i.e. it is not bound by any other $(\rho' \leq \rho, K' \leq K)$), then this bound is "tight."

(Example). The tightest bound on x's visit using this definition is $(\rho = 30s, K = 2)$. Even though e_{leave} is only 10 seconds, it is not possible to choose $\rho < 30$ as this would not contain e_{enter}. To be explicit, x's visit is also (loosely) (ρ, K)-bounded for any $\rho \geq 30s$ and $K \geq 2$.

Definition 3.2.3 ((ρ, K)-neighboring videos)**.** Two video segments v, v' (of the same length and frame rate) are (ρ, K)-neighboring if the set of frames in which they differ[1] is (ρ, K)-bounded.

(Example). Suppose v contains the event e (x entering and leaving the building). One potential v' is a hypothetical video in which x was never present (but everything else observed in v remained the same). Note this is purely to denote the strength of the guarantee in the following definition, no parties need to construct such a v'.

Definition 3.2.4 ((ρ, K)-event-duration ϵ-differential privacy)**.** A randomized mechanism $\mathcal{M}$ satisfies (ρ, K)-event-duration ϵ-differential privacy (abbreviated (ρ, K)-duration privacy) iff for all possible pairs of (ρ, K)-neighboring videos v, v', any finite set of queries $Q = \{q_1, q_2, ...\}$ and all $S_q \subseteq Range(\mathcal{M}(\cdot, q))$:

$$Pr[(\mathcal{M}(v, q_1), \ldots, \mathcal{M}(v, q_n)) \in S_{q_1} \times \cdots \times S_{q_n}] \leq$$
$$e^{\epsilon} Pr[(\mathcal{M}(v', q_1), \ldots, \mathcal{M}(v', q_n))) \in S_{q_1} \times \cdots \times S_{q_n}]$$

Guarantee. (ρ, K)-duration privacy protects all (ρ, K)-bounded events (such as x's visit to the building) with ϵ-DP: informally, if an event is (ρ, K)-bounded, an adversary cannot increase their knowledge of whether or not the event happened by observing a query result from $\mathcal{M}$. To be clear, (ρ, K)-duration privacy is *not* a departure from DP, but rather an extension to explicitly specify what to protect in the context of video.

[1] A pair of frames from the same timestamp of v and v' differ if *any* pixel values are not identical.

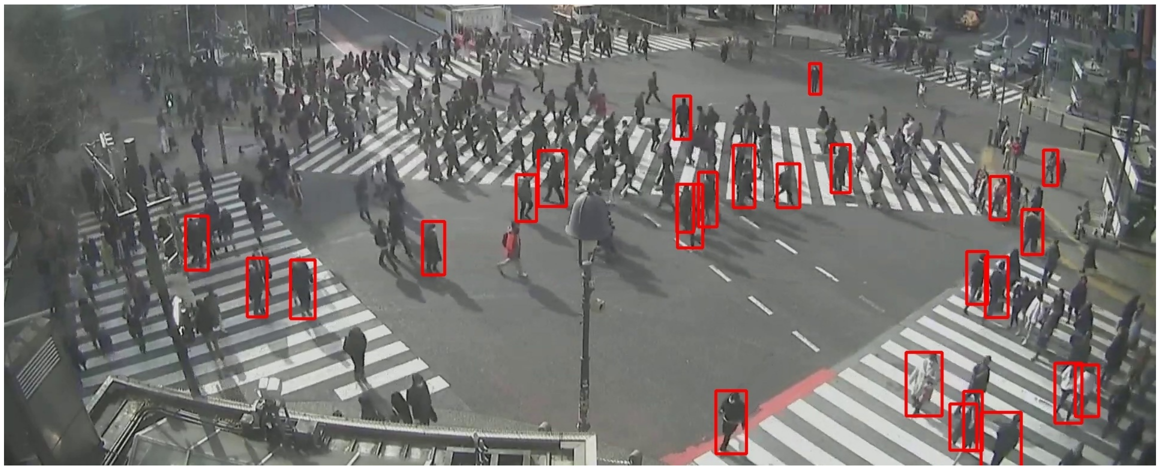


Figure 3-1: The results of a state-of-the-art object detection algorithm (filtered to "person" class) on one frame of `urban`. The algorithm misses 76% of individuals in the frame, but is *still* able to produce a conservative bound on the maximum duration of all individuals (Table 3-1).

Video	**Maximum Duration**		**% Objects CV Missed**
	Ground Truth	CV Estimate	
`campus`	81 sec	83 sec	29%
`highway`*	316 sec	439 sec	5%
`urban`	270 sec	354 sec	76%

Table 3-1: Despite the imperfection of current CV algorithms (exemplified by % objects they failed to detect), they still produce a conservative estimate on the duration of any individual's presence. *For the purposes of this experiment, we ignored cars that were parked for the entire duration of the segment.

3.3 Choosing a Privacy Policy

The Video Owner is responsible for choosing the parameter values (ρ, K) ("policy") that bound the class of events they wish to protect. They may use domain knowledge, employ CV algorithms to analyze durations in past video from the camera, or a mix of both. Regardless, they express their goal to Privid solely through their choice of (ρ, K). Overall, their goal is to choose the tightest possible policy that satisfies their goals. Choosing a loose policy is relatively easy, but will result in an unnecessarily large amount of noise and thus low utility for analysts. The Video Owner wants to maximize the utility of analysts subject to the constraint that its privacy goals are satisfied.

Automatic setting of (ρ, K). The primary reason (ρ, K)-duration privacy is *practical* is that, despite their imperfections, today's CV algorithms are capable of producing good estimates of the maximum duration any individuals are visible in a scene. We provide some evidence of this intuition over three representative videos from our evaluation in Table 3-1. For each video, we chose a 10-minute segment and manually annotated the duration of each individual (`person` or `vehicle`), i.e., "Ground Truth", then used state-of-the-art object detection and tracking to estimate the durations ("CV"). The table reports the maximum duration in each case. The third column ("% Objects CV Missed") is the fraction of objects in our ground that were not identified in at least one frame by the CV algorithm. The key takeaway from these results is that, while object detection misses a non-trivial fraction of bounding boxes, the tracking algorithm is able to fill in the gaps for enough trajectories to capture a "conservative" (i.e., greater than the ground truth) estimate of the maximum duration. In other words, for our three videos, using these imperfect CV algorithms to pick (ρ, K) would successfully capture the duration of, and thus protect the privacy of, *all* individuals, while using the same CV algorithms to implement any prior approach would not.

Relaxing the set of private individuals. Sometimes protecting *all* individuals is unnecessary. Consider a camera in a store; employees will appear significantly longer and more frequently than customers (e.g., 8 hours every day vs. 30 minutes once a week), but if the fact that the employees work there is public knowledge, the Video Owner can pick a policy (with smaller ρ and K) that only bounds the appearance of customers.

Generic policies. Alternatively, the Video Owner can choose a policy to place a generic limit on the (temporal) granularity of queries. Consider a policy ($\rho = 5\text{min}, K = 1$). Suppose individual x stops and talks to a few people on their way to work each morning, but each conversation lasts less than 5 minutes. Although the policy does not protect x's presence or even the fact that they often stop to chat on their way to work, it *does* protect the timing and contents of each conversation.

Time-Varying Policies. Policies are defined over a segment of video from a camera, not an entire camera itself. Thus, different policies can be used for different time ranges of the same camera. For example, when roads are congested in the mornings and evenings, the maximum duration and thus tightest policy may be much higher than night or mid-day when roads are clear and cars can pass freely. Similarly, walkways may be more congested by pedestrians during weekends than weekdays. The more data the Video Owner can analyze, the more of these patterns they can capture and thus exploit to ensure the tightest possible policy for each time period and thus minimum amount of noise for analysts.

The Video Owner can choose to set different policies at any temporal granularity, but if they are too fine-grained an analyst may not be able to take advantage of them; if a query aggregates over a segment of video with multiple policies, in the worst case the event could have occurred during the time range with the loosest policy (i.e., the policy that translates to the largest number of chunks) and thus the loosest policy applies. However, since policies are public information, analysts can take this into account when structuring their queries. For example, consider a camera observing a traffic light, where the light is green for 3 minutes, then red for 1 minute. The Video Owner could set $(\rho = 10s, K = 1)$ during green lights and $(\rho = 1min, K = 1)$ during red lights. However, a query over an entire hour of video would use $(\rho = 1min, K = 1)$ because it is the loosest of the included policies.

Updating Policies. The distribution of durations tends to be a property of a scene, including the buildings and infrastructure present which define how individuals move through the scene and where they spend their time. Intuitively, these properties of a scene do not change over short periods of time (hours, days, or weeks), but may change over longer periods of time (months or years). Thus, the Video Owner should periodically (or continuously) re-run their analysis to monitor the distribution of durations and update the policy as necessary.

3.4 Privacy Guarantee Semantics

In the previous section, we discussed different methods for choosing a privacy policy. A key claim was that this policy need not be "perfect." A privacy policy protects (ρ, K)-bounded events with ϵ-DP, but in practice most events will not be exactly contained by (ρ, K). They may be shorter or longer. They may occur multiple times or be observed by multiple cameras. A key claim of this thesis is that capturing all of these nuances is impractical. Importantly, (ρ, K)-duration privacy provides *some* level of privacy for *all* individuals. In this section we explore what level of privacy individuals are afforded based on how long they are visible.

In short, events that are visible for longer receive proportionally less privacy. However, by definition, events that are visible for longer are coarser, and less specific.

3.4.1 Privacy is Proportional to Duration

A (ρ, K) policy provides a relative reference point: events that exactly match the policy (i.e., made up of *exactly* K segments each of duration ρ) are protected with ϵ-DP, while events that are visible for shorter or longer durations are protected with a proportionally (w.r.t. the duration) stronger or weaker guarantee, respectively.

Theorem 3.4.1. Consider a camera with a fixed policy (ρ, K, ϵ) and an event e. If the tighest bound on e is $(\hat{\rho}, \hat{K})$, then (ρ, K)-duration privacy protects e with $\hat{\epsilon}$-DP, where $\hat{\epsilon} = \frac{\hat{\rho}\hat{K}}{\rho K}$, which grows (degrades) as $(\hat{\rho}, \hat{K})$ increase while (ρ, K, ϵ) are fixed. The actual guarantee provided by PRIVID is a (slightly) looser $\hat{\epsilon} = O(\frac{\hat{\rho}\hat{K}}{\rho K})\epsilon$ (where the constants do not depend on the query) due to the quantization of time by chunking. We state this formally and prove it in §4.3.1.

(Example 1). Given a policy of $(\rho = 1hr, K = 1)$, a single 2-hour appearance would be protected with $\sim 2\epsilon$-DP (weaker) and a single half-hour appearance would be protected with $\sim \frac{1}{2}\epsilon$-DP (stronger).
(Example 2). Given a policy of $(\rho = 30min, K = 2)$, the situation is the same: a single 2-hour appearance would be protected with $\sum 2\epsilon$-DP and a single half-hour appearance would be protected with $\sim \frac{1}{2}\epsilon$-DP.
(Example 3). Suppose Alice and Bob are eating lunch at a restaurant and are observable by a camera with a policy of $(\rho = 60s, K = 1)$. Let us consider some increasingly coarse characterizations of events that comprise this lunch and how long they are "visible:"

- Specific words that are spoken have a duration $<$ 1s (and are thus protected with $\frac{1}{60}\epsilon$-DP).
- Specific actions, sentences and behaviors have a duration on the order of a few seconds (and are thus protected with $\sim \frac{1}{10}\epsilon$-DP).
- Conversation topics have a duration on the order of minutes (and are thus protected with $\sim \epsilon$-DP).
- The mood and sentiment of their conversation may have a duration on the order of tens of minutes (and are thus protected with $\sim 10\epsilon$-DP).
- The event of them meeting together may have a duration of 1-2 hours (and is thus protected with $\sim 60\epsilon$-DP).

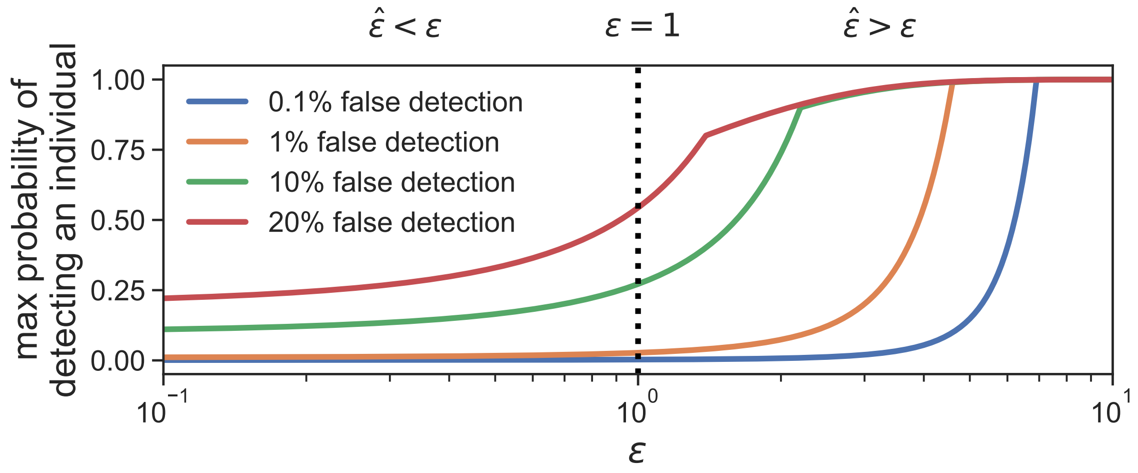


Figure 3-2: Plot of Equation 3.3 for a few different levels of α. Note that the x-axis is plotted for absolute values of ϵ and is using a log scale. The y-axis is the maximum probability that an adversary with a given confidence level could detect whether or not x was present. If one draws a vertical line at the value of ϵ being enforced (e.g., we mark $\epsilon = 1$ here), the trend to the left shows how privacy is improved for individuals who are visible for less time, and the right shows how it degrades for those who are visible for more.

In other words, although this policy does not necessarily prevent an adversary from detecting that Alice and Bob met in general, it does protect most of the details of their meeting.

Graceful degradation. An important corollary of this notion of proportional privacy guarantees is that the level of privacy degrades "gracefully". As an event's $\hat{\rho}$ increases further from ρ (or $\hat{K}$ from K), its effective $\hat{\epsilon}$ increases linearly, yielding a progressively weaker guarantee. (The reverse is true, as $\hat{\rho}$ and $\hat{K}$ decrease, it yields a stronger guarantee). Thus, if $\hat{\rho}$ (or $\hat{K}$) is only *marginally* greater than ρ (or K), then the event is not immediately revealed in the clear, but rather is protected with $\hat{\epsilon}$-DP, which is still a DP guarantee, only marginally weaker: a malicious analyst has only a marginally higher probability of detecting x in the worst case. This in effect *relaxes* the requirement that (ρ, K) be set strictly to the maximum duration an individual could appear in the video to achieve useful levels of privacy. We generalize and provide a visualization of this degradation in the following subsection.

3.4.2 Detection Probability

Although $\hat{\epsilon}$ provides a way to quantify the level of privacy provided to each individual, it can be difficult to reason about relative values of ϵ and what they ultimately mean for privacy in practice. We can use the framework of binary hypothesis testing to develop a more intuitive understanding and ultimately visualize the degradation of privacy as a function of $\hat{\epsilon}$ relative to ϵ.

Consider an adversary who wishes to determine whether or not some individual x appeared in a given video V. They submit a query Q to the system, and observe only the final result, A, which PRIVID computed as $A = Q(V) + \eta$, where η is a sample of Laplace noise as defined in the previous section. Based on this value, the adversary must distinguish between one of two hypotheses:

$$\begin{aligned}\mathcal{H}_0 &: x \text{ does not appear in } V\\ \mathcal{H}_1 &: x \text{ appears in } V\end{aligned}$$

We write the false positive P_{FP} and false negative P_{FN} probabilities as:

$$\begin{aligned}P_{FP} &= \mathbb{P}(x \in V | \mathcal{H}_0)\\ P_{FN} &= \mathbb{P}(x \notin V | \mathcal{H}_1)\end{aligned}$$

From Kairouz [44, Theorem 2.1], if an algorithm guarantees ϵ-differential privacy ($\delta = 0$), then these probabilities are related as follows:

$$P_{FP} + e^{\epsilon} P_{FN} \geq 1 \tag{3.1}$$

$$P_{FN} + e^{\epsilon} P_{FP} \geq 1 \tag{3.2}$$

Suppose the adversary is willing to accept a false positive threshold of $P_{FP} \leq \alpha$. In other words, they will only accept $\mathcal{H}_1$ (x is present) if there is less than α probability that x is not actually present.

Rearranging equations 3.1 and 3.2 in terms of the probability of correctly detetecting x is present $(1 - P_{FN})$, we have:

$$\begin{aligned}1 - P_{FN} \leq && e^{\epsilon} P_{FP} \leq && e^{\epsilon}\alpha\\ 1 - P_{FN} \leq && e^{-\epsilon}(P_{FP} - (1 - e^{\epsilon})) \leq && e^{-\epsilon}(\alpha - (1 - e^{\epsilon}))\end{aligned}$$

Then, for a given threshold α, the probability that the adversary *correctly* decides x is present is *at most* the minimum of these:

$$\mathbb{P}(x \in V | \mathcal{H}_1) \leq \min\{e^{\epsilon}\alpha, e^{-\epsilon}(\alpha - (1 - e^{\epsilon}))\} \tag{3.3}$$

In Figure 3-2, we visualize 3.3 as a function of ϵ for 4 different adversarial confidence levels (α=0.1%,1%,10%,20%). As an example of how to read this graph, suppose Privid uses a $(\rho = 60s, K = 1, \epsilon = 1)$ policy ($\epsilon = 1$ marked with the dotted line). An individual who appears 3 times for $< 60s$ each is $(\rho = 60s, K = 3)$-bounded, and thus has an effective $\hat{\epsilon} = 3$ relative to the actual policy for most queries (Theorem 3.4.1). If an adversary has a $\alpha = 1\%$ confidence level, then they would have at most a $\sim 20\%$ chance of correctly detecting the individual appeared, even though they appeared for far more than the policy allowed. We can also see that, for sufficiently small values of ϵ (e.g., $\epsilon < 1$), even if the adversary has a very liberal confidence level (say, 20%), a marginal increase in $\hat{\epsilon}$ relative to ϵ only gives the adversary a marginally larger probability of detection than they would have had otherwise.

An important takeaway is that, when an individual exceeds the (ρ, K) bound protected by Privid, their presence is not immediately revealed. Rather, as it exceeds the bound further, $\hat{\epsilon}$ increases, and it becomes more likely an adversary could detect the event.

3.4.3 Multiple Appearances

The larger the time window of video a query analyzes, the more instances an individual may appear within the window, even if each appearance is itself bounded by ρ. Consider our example individual x and policy $(\rho = 30s, K = 2)$ from §3.2. In the query window of a single day d, x appears twice; they are properly (ρ, K)-bounded and thus the event "x appeared on day d" is protected with ϵ-DP. Now, consider a query window of one week; x appears 14 times (2 times per day), so the event "x appeared sometime this week" is $(\rho, 7K)$-bounded and thus protected with (weaker) 7ϵ-DP. However, the more specific event "x appeared on day d" (for any d in the week) is *still* (ρ, K)-bounded, and thus still protected with the same ϵ-DP. In other words, while an analyst may learn that an individual appeared *sometime* in a given week, they cannot learn on which day they appeared. Thus, in order to get greater certainty, the analyst must give up temporal granularity.

3.4.4 Multiple Cameras

When an individual appears in front of multiple cameras, their privacy guarantees are analogous to the previous case of repeated appearances in a single camera. If they appear in front of N different cameras, where the event of their appearance in camera i is protected with $\hat{\epsilon}_i$-DP, then the event of their appearance across all the cameras is protected with $\sum_i \hat{\epsilon}_i$-DP. Suppose they appear in front of some 10 cameras in a large (1000 camera) network, and that $\sum \hat{\epsilon}_i$ is large enough for the adversary to detect their appearance with high confidence. Then while the adversary could infer the person appeared *somewhere* in the camera network, the adversary would not learn *which* cameras they appeared in or when; appearances within individual cameras are still protected by ϵ-DP.

Limitation. One caveat of the above is when a set of cameras can be used to infer a semantically-meaningful trajectory for an individual, *and* the adversary knows this. For example, consider the route from one's home to work. If there are 10 cameras along this route, and the individual is unlikely to take this route to reach a different destination, then a query indicating they were present in a majority of these 10 cameras could be enough to infer that they took the trip to work. In other words, even though our guarantee (as described above) protects exactly which camera at exactly which time, the correlation between cameras allows the adversary to infer more. In this case, the more cameras uniquely associated with a particular trajectory, the more confidence an adversary could gain about the trajectory. Solving this situation is difficult and would be an important direction for future work (see the challenges described in §3.5.4).

3.4.5 Multiple Targets

DP defines a worst-case guarantee for a *single* individual. In other words, in the worst case, if an analyst uses the entire available query budget ϵ to pose a query about the presence of some event or person, X, they necessarily cannot pose another query about the presence of some other event, Y. If they want to learn about both X and Y, they must split the budget between these two queries. The more events they wish to target, the more they must split the budget. This naturally inhibits surveillance. Our ideal goal is to ensure that it is not possible to target even a single person. However, surveilling multiple people is proportionally more difficult and less feasible or accurate. Even in cases where it is possible to learn some information about one person, it is likely infeasible to learn much significant information about multiple people. Further, DP bounds the worst-case privacy leakage in the event that an adversary specifically targets an individual. However, if they do not (or are not able to), then the actual privacy leakage may be less (or zero). These properties are not unique to our notion of privacy, they are inherited from DP.

3.5 Alternative Formulations

In this section we explain why we chose the particular formulation of the definition as opposed to some similar alternatives.

3.5.1 Why specify K?

If the video is split into individual frames, K is unnecssary. Consider a window of 100 frames of video, split into individual frames. Whether an event appears in frames 1 and 2 or 1 and 100, it ultimately occupies a total of 2 frames. This can be captured by ρ. If we wanted to protect two of these events, we could just pick a policy of 2ρ. However, a key practical requirement of Privid is the ability to operate over chunks of frames to execute queries that require (temporally) local state. When using chunks, time is quantized. If chunks are 10 frames, for example, an event appearing in frames 1 and 2 is spanned by a single chunk, while an event appearing in frames 1 and 100 is spanned by two chunks. In other words, there is a difference between consecutive and disjoint events when dealing with chunks. K allows us to specify when we want to protect disjoint events so that our sensitivty calculation can account for them properly.

3.5.2 Why not define ρ as the total time?

In (ρ, K)-duration privacy, ρ is the length of *each* of the K segments. We could instead define ρ as the *total* duration, distributed arbitrarily across K segments. However, this bears less resemblance to real events and results in a far looser bound when we quantize time based on chunks in our implementation of Privid: in the worst case, the event could be divided into $\rho \times$ fps segments of single frames, each appearing in a different chunk. As the chunks get longer, this allows a single event to span more and more chunks. While this may not seem realistic, it would be allowed by the definition. Alternatively, we could specify the length of each of the K segments, rather than assigning them all the same length ρ. However this quickly becomes complicated and more difficult to reason about. While there is nothing fundamentally preventing either of these formulations, we felt that our particular formulation struck the right balance between practicality and interpretability.

3.5.3 Why not define policies relative to window size?

One formulation we initially considered was a privacy policy that was a function of the query window size. This was motivated by the recognition that recurring events, such as an individual entering and leaving their workplace, can be observed more by aggregating over a larger window. Over a single day, the event occurs twice, but over a month, the event occurs 40 times (ignoring weekends). If an adversary aggregates over an entire month, a standard $(\rho, K = 2, \epsilon)$ policy would protect an individual on any one day, but would reveal that they frequently went to work during that month. In this case, we can semantically capture the desired privacy goal by defining the number of events per time period. In this case, we want to protect 2 events *per day*. Then, if a query aggregated over a single day, the policy would protect 2 events as normal with ϵ-DP, but if they aggregated over a month, the policy would dictate that the set of 40 events should be protected with ϵ-DP. This is equivalent to protecting each individual event with the much stronger $\frac{1}{40}\epsilon$-DP.

Unfortunately, this fundamentally does not work. The analyst could issue a separate query for each day of the month, which protects each day with ϵ-DP. They could then average these results (query composition) to get a result that satisfies only 40ϵ-DP, which is not strong enough to prevent detection.

3.5.4 Why not provide a guarantee over multiple camearas?

Given the limitations described in §3.4.4, it is tempting to scale the privacy definition based on the number of cameras included in the query aggregation. For example, in the case of a unique trajectory captured by a set of camears, adding noise proportional to the number of cameras would protect the existence of that trajectory. However, this fundamentally does not work for the same reason we cannot scale based on the window size (§3.5.3): the analyst could issue a query over each of the cameras observing the trajectory independently then average out the noise, which would defeat the purpose. Thus, malicious queries could avoid the restriction, and it would only penalize benevolent queries that have a genuine need for aggregating across multiple cameras in one query.

Chapter 4

Privid: A System for Event-Duration Privacy

In this chapter, we present Privid, a general-purpose video analytics system that both (a) satisfies (ρ, K)-duration privacy (§3) and (b) provides an expressive query interface which allows analysts to supply their own (untrusted by Privid) video-processing code.

Privid does not take a particular stance on *what* is private, it simply satisfies the definition given to it by the Video Owner. It also contains a wide number of knobs for analysts to customize their queries. The goal of this chapter is to explain how Privid works, given a particular policy from the Video Owner and query from the analyst. In Chapter 6 we step back and discuss how Video Owners and analysts can make informed decisions about these parameters in practice.

In the first section (§4.1), we provide a birds-eye view of the system and all of its components. The following sections elaborate on each of these components in greater detail. We evaluate Privid in Chapter 5.

4.1 Overview

4.1.1 Threat Model

Recall from §1.2 that the Video Owner does not trust the analyst enough to freely share their video streams. Instead, they employ Privid as an intermediate layer between the raw video and the analysts. Privid retains full control over the video data, analysts can only extract inferences from the video via Privid's query interface. Privid is responsible for executing the query and adding noise such that the result satisfies (ρ, K)-duration privacy. It returns only the result to the analyst. The only side channel available to the analyst is the time they receive the result. PRIVID protects this side channel, which we describe in §4.2.4.

The Video Owner does not place any trust in the intentions of the analysts or any of their query processing code. Analysts pose queries to PRIVID adaptively: the full set of queries is not known ahead of time and analysts may use the results of prior queries when posing new ones. Any number of analysts may be malicious and may collude to violate the privacy of the same individual.

However, we assume analysts trust the Video Owner and PRIVID to faithfully execute their query, and they are willing to share their processing code so that the Video Owner can execute it. This trust requirement could be alleviated by deploying PRIVID in the cloud and using a trusted execution environment tailored to video analytics, such as Visor [70].

Policies. PRIVID does not guarantee a particular semantic definition of privacy. Rather, the Video Owner picks a policy (ρ, K) and privacy budget (ϵ) for each camera they manage that matches their desired semantic level of privacy. Given these parameters, PRIVID provides a guarantee of (ρ, K)-duration privacy (Theorem 4.4.1) for all queries over all cameras it manages.

Metadata. The only other information released by PRIVID aside from query results is fixed per-camera metadata (described in §6.1), which is made publicly available to analysts to guide their query creation. This includes the policy, the available budget, and details about the camera itself. PRIVID does not provide any privacy guarantees about this information. However, this information is manually supplied by the Video Owner, fixed a priori, and is not a function of the video data that the analyst queries, so any potential privacy leaks are clearly interpretable and can be minimized.

4.1.2 Query Support

In particular, PRIVID supports *aggregation* queries, which process a "large" amount of video data (e.g., several hours/days of video) and produce a "small" number of bits of output (e.g., a few 32-bit integers). Examples of such tasks include counting the total number of people that passed by a camera in one day, or computing the average speed of cars observed. In contrast, PRIVID does not support a query such as reporting the location (e.g., bounding box) of a particular person or car within the video frame. PRIVID can be used for one-off queries or standing queries running over a long period, e.g., the total number of cars per day over a year.

4.1.3 Execution Model

Privid requires queries to be expressed using a *split-process-aggregate* model, where (1) the video is split (temporally) into chunks, (2) each chunk is processed to form rows of a table, and (3) a SQL-like aggregation query is run over the table to produce a raw result. The analyst has control over each stage of this pipeline, which allows analysts to express a wide range of queries. While video analytics queries are typically constructed this way implicitly, by forcing queries to be expressed this way explicitly, Privid provides the following primitive regarging the aggregate result: *the duration an event is visible to a camera is directly proportional to the amount it can impact the result.* This primitive is the key piece that allows Privid to compute a reasonable bound on the sensitivity of a query, which is necessary to satisfy DP. Privid adds noise to queries using the standard Laplace mechanism, scaled according to the sensitivity of the query.

4.1.4 Components

Algorithm 1 provides an overview of the entire Privid mechanism in pseudocode.

The rest of the chapter is structured as follows. In §4.2 we describe Privid's domain specific query language. In §4.3 we explain how Privid computes the sensitivity of a query and prove that it satisfies our definition of (ρ, K)-duration privacy. In §4.4 we explain how Privid ensures that queries compose safely, i.e. that the privacy guarantee is upheld even when many queries are posed over the same video.

The noise that Privid adds to a query result is proportional to both: (a) the privacy policy (ρ, K), which determines the number of chunks an individual can occupy, and (b) the query's constraints, which determine how much each chunk can impact the aggregate output. Privid includes two optional components to improve query accuracy for analysts while maintaing an equivalent level of privacy for Video Owners. The first, spatial masking (§4.5), enables a tighter privacy policy, while the second, spatial splitting (§4.6) enables tighter constraints on the range of each chunk.

4.2 Query Interface

4.2.1 Query Contents

A Privid query must contain (1) a block of statements in a SQL-like language, which we introduce below and call PrividQL, and (2) video processing executables.

Algorithm 1: PRIVID Mechanism

```
   Input  : PRIVID query $Q = \{[F...], [S...], c, \epsilon_Q\}$, videos $V$, policy $(\rho, K, \epsilon)$
   Output: Query answer(s) $A$
   // Ensure sufficient budget for entire interval and $\rho$ margin for
      all aggregations
1  foreach $s \in S$ do
2      foreach $f \in s.V[s.I \pm \rho]$ do
3          if $f.budget < \epsilon_Q$ then
4              return DENY
   // There is enough budget for all aggregations, so query is
      permitted, decrement budget for entire interval for all
      aggregations
5  foreach $s \in S$ do
6      foreach $f \in s.V[s.I]$ do
7          $f.budget$ -= $\epsilon_Q$
   // Create intermediate tables $T$
8  foreach $c \in C$ do
9      itable $\leftarrow$ Table(c.output_schema)
10     chunks $\leftarrow$ Split $V[I]$ into sequential segments each of length
        c.chunk_sec with stride c.chunk_stride_sec
11     foreach $chunk \in chunks$ do
12         rows $\leftarrow$ F[c.process_using](chunk) // Executed in confined
              environment
13         itable.append(rows)
14     tables[c.table_name] $\leftarrow$ itable
   // Compute output and add noise
15 foreach $(i, s) \in S$ do
16     $r \leftarrow$ execute SQL query $s$ over table(s) in $T$, includes joins etc.
17     $\Delta_{(\rho,K)} \leftarrow$ compute recursively over $s$, using rules in Table 4-3 Equation 4.2
18     $\eta \leftarrow Laplace(\mu = 0, b = \frac{\Delta}{\epsilon_Q})$
19     $A_i \leftarrow r + \eta$
```

(1) PrividQL statements. A valid PQL block involves one or more of *each* of the 3 following statements:

• `SPLIT` statements choose a segment of video (camera, start and end datetime) as input, and produce a set of video chunks as output. They specify how the segment should be split into chunks, i.e., the chunk duration and stride between chunks.

• `PROCESS` statements take a set of `SPLIT` chunks as input, and produce a traditional ("intermediate") table. They specify the executable that should process the chunks, the schema of the resulting table, and the maximum number of rows a chunk can output (maxrows, necessary to bound the sensitivity, Equation 4.1). Any rows output beyond the max are dropped.

• `SELECT` statements resemble typical SQL `SELECT` statements that operate over the tables resulting from `PROCESS` statements. However, they must have an aggregation as the final operation, and they must supply some additional constraints on the values they operate over (described in Figure 4-3). Privid supports the standard aggregation functions (e.g., `COUNT`, `SUM`, `AVG`) and the core set of typical operators as internal relations. Each `SELECT` constitutes a separate data release, and thus receives its own sample of noise and consumes some privacy budget. The only special case is a `SELECT` with a `GROUPBY` (time), which executes the same query over subsequent time ranges (e.g. a count per hour). This is just syntactic sugar for a separate query for each hour, and thus each group is a separate data release, receive its own sample of noise, and consumes privacy budget. In order to aggregate across multiple video sources (separate time windows and/or multiple cameras), the query can use a `SPLIT` and `PROCESS` for each video source, and then aggregate using a `JOIN` and `GROUPBY` in the `SELECT`.

(2) `PROCESS` executables. Analysts must also attach the executable(s) used by the `PROCESS` statements in their query. Executables take one chunk as input, and produce a set of rows (e.g., one per object) as output.

In practice, it does not matter whether these statements are provided within the same request, or separate ones. The system could receive request A which creates chunks AC and table AT and then aggregates over AT to release result AR. A future request B could also aggregate over table AT to release a different result BR. From the system's perspective, for the purpose of sensitivity calculation, this is equivalent to a "single" query involving chunks AC, table AT, and result BR.

```
query := split_stmt | process_stmt | select_stmt
split_stmt := SPLIT camera_id
    BEGIN timestamp
    END timestamp
    BY TIME chunk_sec STRIDE stride_sec
    [BY REGION ...] // optional, see Section 7
    [WITH MASK ...] // optional, see Section 7
    INTO chunk_set_id;
process_stmt := PROCESS chunk_set_id
    USING binary_name
    TIMEOUT timeout_sec
    PRODUING maxrows
    WITH SCHEMA /* list of */ column_schema
    INTO table_id;
select_stmt := outer_select FROM inner_select
    [GROUP BY col_list WITH KEYS ...]
outer_select := SELECT agg_fun(col_name)
inner_select := table_id | process_stmt
    | SELECT expr_list FROM inner_select
        [WHERE condition] [LIMIT rows]
    | inner_select GROUP BY col_list [WITH KEYS ...]
    | inner_select JOIN inner_select ON col_list
column_schema := col_name:dtype=default
agg_fun := SUM | COUNT | AVG | ...
expr := col_name | expr + expr | expr * expr | ...
dtype := STRING | NUMBER
```

Figure 4-1: PRIVID Query Grammar. Terms in capital letters are query language keywords. Keywords in square brackets are optional. The term `col_name` stands for the name of an analyst-provided column.

4.2.2 PRIVID Query Language (PQL)

In this section, we provide a more in-depth description of PRIVID's query language, PQL. This language is a domain-specific version of SQL, which is intentionally as close to SQL as possible to be friendly to analysts. Figure 4-1 contains the full grammar.

A valid PQL statement contains at least one `split_stmt`, followed by at least one `process_stmt`, and finally at least one `select_stmt`.

A `split_stmt` converts a segment of video data from a single camera into a named set of chunks by specifying the following:

- The `BEGIN` and `END` timestamps describe the bounds of time the analyst is interested in. Tables are evaluated lazily only once they are needed for an aggregation so the analyst can choose large time bounds (e.g.,, an entire year) but narrow to specific times (e.g.,, 1 hour per weekday) using the aggregation statement. These times may be in the past or future (i.e., for streaming queries). Any values that only depend upon past timestamps will be processed and released as soon as possible (limited only by the processing time requirements described in §4.2.4). Any values that depend upon future timestamps will be released as soon as possible (given the timeout) after all of the timestamps needed have elapsed.

- `BY TIME` describes the duration of each chunk, and the `STRIDE` between chunks. Both values must correspond to an integer number of frames (e.g., at a frame rate of 30 fps, 0.5 seconds is permitted because it corresponds to 15 frames, but 0.25 seconds is not permitted because it corresponds to 7.5 frames). The chunk duration must be positive. The stride defaults to matching the chunk duration (sequentual non-overlapping chunks), but can be less than the chunk duration (for overlapping chunks) or greater than the chunk duration (to intentionally skip space between chunks).

- `BY REGION` describes the scheme used to further split each chunk spatially. The set of possible schemes is determined manually by the Video Owner and made available as part of the public camera metadata.

- `WITH MASK` specifies the id of a Video Owner-provided mask. A mask specifies a set of pixels to remove from the video (i.e., replace with black pixels). This mask is applied to the video before splitting, and thus before the analyst's executable is able to view the video.

A `process_stmt` uses the analyst-provided executable to convert a set of chunks (created by a `split_stmt`) into an intermediate table by specifying the following:

- `USING` provides the path of the analyst-provided executable that should be used to process each chunk of this camera's video data. Analysts may supply any number of executables and use different executables for different cameras. These executables take as input a list of (contiguous) frames from the video (a "chunk"), and output rows of a table (whose schema is defined by the `PRODUCING` directive). Each chunk is processed by an independent instantiation of the executable in a confined execution environment (§4.2.5).

- `TIMEOUT` specifies the maximum amount of time that can be used to process each chunk. If execution exceeds this time for any chunk, it is immediately terminated and a row is output with the `default` values for each column as specified in the `user_schema`. The existence of the `TIMEOUT` clause is crucial for preventing side-channel information leakage via the execution time (§4.2.4).

- `PRODUCING` *`maxrows`* `WITH SCHEMA` *`schema`* specifies the schema of columns in the table and the maximum number of rows each chunk will output. For each column, the schema specifies a name (for reference in aggregations), a data type (either `STRING` or `NUMBER`, used to determine the types of aggregations permitted over the column), and a default value (to be output if the processing for that chunk crashes or exceeds `TIMEOUT`). Privid does not place any trust in the executable or make any assumptions about the content of the output; it truncates the output as necessary to ensure that it adheres to the schema.

  In addition to the user-specified columns, Privid also adds a `chunk` column to every table which contains the timestamp of the first frame of the chunk. This can be used to narrow time ranges (e.g., only 12pm-2pm each weekday), aggregate over different amounts of time (e.g., group results per hour), or match times across cameras or days.

- `INTO` *`table_id`* specifies the name of the resulting table. The `PROCESS` directive always creates a *new* table, it cannot add rows to an existing table. Tables can be appended to each other, but this must be done in the `SELECT` statement, so the impact of the append on the sensitivity can be computed properly.

Table Selection and Aggregation. A selection-aggregation statement `select_stmt` computes aggregate statistics from intermediate tables (produced by `process_stmts`) using familiar SQL syntax, with some important restrictions to properly control sensitive data leakage:

- The outer-most select (`outer_select`) must be an aggregation. Each aggregation must be over a single column (with the exception of `COUNT(*)`) and is treated as an independent result r_i. Privid uses the Laplace mechanism to add an independent sample of noise to each r_i before releasing it to the analyst, and subtracts from the privacy budget for each r_i as well. The select can optionally group results using a `GROUP BY`, but only if it explicitly provides the keys (using `WITH KEYS [...]`, so that they are not dependent on the data) or groups over the `chunk` column (which Privid created and therefore can trust). Figure 4-3 lists the supported aggregation functions and some restrictions.

- An `inner_select` statement is nested inside an `outer_select` statement and can be nested inside other `inner_select` statements. An `inner_select` may transform the original table into a new one, combine multiple tables, and select and project rows and columns.

- Some aggregation functions require the range of a column or the number of rows to be constrained (Figure 4-3). When these cannot be inferred automatically, they must be explicitly provided by the analyst as part of the select via the `range(col, low, high)` function or the `LIMIT rows` directive, respectively.

- Privid includes helper functions, such as `hour(chunk)` or `day(chunk)`, which convert the chunk timestamp into the corresponding hour or day. We note their existence simply because they make queries much easier to read.

- The `CONSUMING` directive specifies how much ϵ budget the analyst wants to use for this aggregation. This budget will be subtracted from all frames that the select uses to produce a result. If there is insufficient budget, the query will be rejected.

4.2.3 Query Compiler

The query compiler converts each select command into an abstract syntax tree with the table at the bottom, inner selects in the middle that transform the table, and finally an aggregation at the top-most layer. The sensitivity engine computes sensitivity recursively starting from the tables at the leaves. While some variables can be unconstrained temporarily during the intermediate layers, all variables must be constrained at the root when the sensitivity of the aggregation function is computed, otherwise the query is rejected.

4.2.4 Query Executables

Since Privid assumes it cannot trust the analyst's executable, it enforces the privacy guarantee using constraints during the execution process.

When running a Privid query, an analyst can observe only two pieces of information: (1) the query result, and (2) the time it takes to receive the result.

Query result. In order to link an event's duration to its impact on the output, Privid ensures that the output of processing a chunk i can *only* be influenced by what is visible in chunk i (not any other chunk j). Then, an individual can *only* impact the outputs of chunks in which they appear, and the duration of their appearance is directly proportional to their contribution to the output table.

To achieve this, Privid processes each chunk using a separate instance of the analyst's executable, each running in its own isolated environment. This environment enforces that the executable can read *only* the video chunk, camera metadata, and a random number generator, and can output *only* values formatted according to the `PROCESS` schema. However, the executable may use arbitrary operations (e.g., custom ML models for CV tasks).

Execution time. To prevent the execution time from leaking any information, we must add two additional constraints. First, each chunk must complete and return a value within a pre-determined time limit T, otherwise a default value is returned for that chunk (both T and the default value are provided by the analyst at query time).[1] Second, Privid only returns the final aggregated query result after $|chunks| \cdot T$. By enforcing these constraints, the observed return time is only a property of the query itself, not the data.

4.2.5 Execution Environment

Formal Specification. Formally, this environment can be modeled as a turing machine with the following properties (where R, T, `schema`, and `default` are specified a priori as part of the query $\mathbb{Q}$):

- Takes as **input**: a set of frames representing a chunk, timestamp of the first frame, frame rate (in fps), camera ID, and (optionally) any additional metadata the Video Owner wishes to provide that is not dependent on any private information, such as the amount of daylight at that time.
- Has access to a **random tape** that is uncorrelated with any of the other chunks
- Produces as **output** at most R rows, each with columns specified by `schema`.
- Executes for exactly T seconds. If it finishes early, it must wait until T seconds have elapsed. If it does not finish in time or crashes, it produces a `default` value.

[1]Timeouts can impact query accuracy, hence analysts should first profile their code to select a conservative limit T.

Implementation. In this section, we list the core components used by our prototype runtime to isolate analyst executables. We believe this environment satisfies the formal specification requirements.

1. Our runtime uses `chroot` to set the root directory for the executable to be an empty directory, so that (1) it cannot view any other files in the system and (2) it is free to write temporary files during its execution. While this is not strictly necessary, it makes it makes the environment much more friendly to existing libraries, e.g., PyTorch and Tensorflow, which depend on creating many temporary files. Once the executable finishes processing this chunk, this fake root directory (and any files created inside it) are removed.

2. Our runtime uses `AppArmor` with a barebones policy (shown in Figure 4-2) that only whitelists a few particular system files and devices needed for common operations. Anything not explicitly listed in the policy is denied, including most system devices, /proc, network sockets, IPC mechanisms, etc. Note that we assume `/dev/urandom` has sufficient entropy that a future value cannot be predicted. If this cannot be guaranteed, an AppArmor rule can be used to map `/dev/urandom` to `/dev/random` (or another source of cryptographically secure randomness).

3. Our runtime executes each instance within its own `namespaces`, giving it the impression that it is the only process on the system, run by the only user, and providing another layer of defense (on top of `AppArmor`) preventing it from access any IPC mechanisms or network interfaces.

4. Our runtime uses `cgroups` to ensure that one instance of the executable cannot communicate with another by exhausting system resources.

5. Our runtime disables (by unsetting the relevant hardware flags to make them privileged) the following CPU instructions that could be used to mount side channel attacks such as those described in [69, 66]: `RDTSC`, `RDPMC`, `RDTSCP`, `RDMSR`, `WRMSR`, `RDPKRU`, `WRPKRU`, `UMONITOR`, `RDPID`. We checked that other instructions capable of creating side channels (e.g., `RDMSR` which can check CPU temperature) are already privileged and thus not accessible to the executable.

6. Our runtime processes chunks serially, as opposed to in parallel. While we are not aware of any side channels that exist given the above combination of

```
/privid/runtime {
  # this /tmp is only visible to this execution and
  # is deleted immediately after execution
  /tmp/**       rwalk,
  # these are frequently used and safe
  /dev/null     rw,
  /dev/zero     rw,
  /dev/random   r,
  /dev/urandom  r,

  # glibc's *printf protections read the maps file
  @{PROC}/@{pid}/{maps,auxv,status} r,
  # glibc malloc (man 5 proc)
  @{PROC}/sys/vm/overcommit_memory  r,

  # allow using common libraries
  # (from default apparmor policy)
  /{usr/,}lib{,32,64}/*                  r,
  /{usr/,}lib{,32,64}/**.so*            mr,
  /{usr/,}lib/@{multiarch}/**           r,
  /{usr/,}lib/@{multiarch}/**.so*      mr,

  # allow this exe to be traced and sent signals
  ptrace (readby),
  ptrace (tracedby),
  signal (receive) peer=unconfined,
  # allow exe to signal itself
  signal peer=@{profile_name},
}
```

Figure 4-2: The AppArmor policy used by our runtime. Each line has the form "`{path} {permissions}`". For the permissions, "r" and "w" refer to read and write access, while "m" means "memory-mapped executable". Any file paths not explicitly listed in this policy cannot be accessed.

mechanisms, this automatically prevents a wide class of side channels that can only succeed when both processes are active.

Dependencies. Privid expects `PROCESS` executables to be standalone executables. They must entirely embed any libraries or ML models they use. We wrote all `PROCESS` executables in our evaluation using Python, and in particular used the `detectron2` [38] library (which itself internally uses `pytorch` [68]) for ML models. We then used `cython3` to create a standalone executable that worked successfully in the constrained runtime environment.

```
> cython3 --embed -o model.c model.py
> gcc -I /usr/include/python3 model.c -lpython3 -o model
```

Performance. All `PROCESS` executables in our evaluation were able to run successfully within this confined environment, and our benchmarks did not indicate any statistically significant overhead relative to the amount of time necessary to process an entire chunk of frames.

Public Availability. The exact implementation that will be used by the system must also be made available to analysts to ensure their executables run properly before submitting a query.

4.3 Query Sensitivity

The sensitivity of a Privid query is the maximum amount the final query output could differ given the presence or absence of any (ρ, K)-bounded event in the video. This can be broken down into two questions: (1) what is the maximum number of rows a (ρ, K)-bounded event could impact in the analyst-generated intermediate table, and (2) how much could each of these rows contribute to the aggregate output. We discuss each in turn.

Contribution of a (ρ, K) event to the table. An event that is visible in even a single frame of a chunk can impact the output of that chunk arbitrarily, but due to Privid's isolated execution environment, it can *only* impact the output of that chunk, not any others. Thus, the number of rows a (ρ, K)-bounded event could impact is dependent on the number of chunks it spans (an event spans a set of chunks if it is visible in at least one frame of each).

In the worst case, an event spans the most contiguous chunks when it is first visible in the last frame of a chunk. Given a chunk duration c (same units as ρ) a single event segment of duration ρ can span at most $\mathsf{max\_chunks}(\rho)$ chunks:

$$\mathsf{max\_chunks}(\rho) = 1 + \lceil \frac{\rho}{c} \rceil \tag{4.1}$$

DEFINITION 4.3.1 (Intermediate Table Sensitivity)**.** Consider a privacy policy (ρ, K), and an intermediate table t (created with a chunk size of c_t in the `SPLIT` and maximum per-chunk rows $\mathsf{maxrows}_t$ in the `PROESS`). The *sensitivity* of t w.r.t (ρ, K), denoted $\Delta_{(\rho,K)}$, is the maximum number of rows that could differ given the presence or absence of any (ρ, K)-bounded event:

$$\Delta_{(\rho,K)}(t) \leq \mathsf{maxrows}_t \cdot K \cdot \mathsf{max\_chunks}(\rho) \tag{4.2}$$

Proof. In the worst case, none of the K segments overlap, and each starts at the last frame of a chunk. Thus, each spans a separate $\mathsf{max\_chunks}(\rho)$ chunks (Eq. 4.1). For each of these chunks, all of the $\mathsf{maxrows}$ output rows could be impacted. □

Sensitivity propagation for (ρ, K)-bounded events. Prior work [58, 43, 49] has shown how to compute the sensitivity of a SQL query over *traditional* tables. Assuming that queries are expressed in relational algebra, they define the sensitivity recursively on the abstract syntax tree, where: the tables are the leaves, the relational operators which transform those tables are the intermediate nodes, and the aggregation function (over the transformed tables) is at the root. Beginning with the maximum number of rows an individual could influence in the input table, they provide rules for how the influence of an individual propagates through each relational operator and ultimately impacts the aggregation function.

Unlike prior work on propagating sensitivity recursively, the intermediate tables in PRIVID are untrusted, and thus require careful consideration to ensure the privacy definition is rigorously guaranteed. In this work, we determined the set of operations that can be enabled over PRIVID's intermediate tables, derived the sensitivity for each, and proved their correctness. Many rules end up being analogous or similar to those in prior work, but `JOIN`s are different. We provide a brief intuition for these differences below. Figure 4-3 contains the complete definition for sensitivity of a PRIVID query.

Each aggregation operation is executed over some inner relation R. The simplest possibility is a selection of a single column from a base table directly, but it could also be over a base table transformed by some inner relational operators. Either way, the sensitivity of the aggregation is defined as a function of the range of the column and the size of R. The size and range must be "constrained" explicitly, i.e. they must have a fixed value that does not depend on the video data. When this is not known directly, analysts must explicitly supply it. PRIVID's query compiler checks to ensure that the size and range are constrained before accepting a query. If not, the query is rejected. There is no harm to the analyst for submitting queries that are rejected, and the process of rejecting a query does not leak any information about the video data or use up any budget.

Privacy semantics of untrusted tables. As an example, consider a query that computes the size of the intersection between two cameras, `PROCESS`'d into intermediate tables t_1 and t_2 respectively. If $\Delta(t_1) = x$ and $\Delta(t_2) = y$, it is tempting to assume $\Delta(t_1 \cap t_2) = \min(x, y)$, because a value needs to appear in both t_1 and t_2 to appear in the intersection. However, because the analyst's executable can populate the table arbitrarily, they can "prime" t_1 with values that would only appear in t_2, and vice versa. As a result, an event need only appear in either t_1 or t_2 to impact the intersection, and thus $\Delta(t_1 \cap t_2) = x + y$.

Why not use an existing DP library? As described earlier, the idea of computing the sensitivity of a SQL query and adding noise to the result is not novel to PRIVID. While there are some libraries that are capable of performing this process, there are a few details of PRIVID that prevent us from using any out-of-the-box. Thus, we derived all of the necessary requirements and properties in detail. If a general-purpose DP implementation arises in the future, PRIVID should be implemented to use this rather than roll its own implementation, as DP (or any privacy/security process) is especailly prone to subtle bugs.

- Per-frame ϵ budget (as opposed to a single budget for the entire dataset). Supporting this requires both maintaining an efficient sparse data structure for the remaining budget across all frames and efficiently checking it for each query.

- Unconstrained variables. Since PRIVID allows the analyst to populate the intermediate table using their own untrusted code, the results cannot be blindly trusted. The query itself must explicitly constrain the variables (e.g., the range of a column), so that PRIVID can compute the appropriate noise. Thus, it must be capable of tracking and propagating unconstrained variables and ensuring that variables have been constrained.

- c-group DP. PRIVID supports standard aggregation functions, but the sensitivity of those aggregations depends upon how much a (ρ, K) event could impact the aggregation's child relation.This is equivalent to ensuring c-group DP over that child relation, where c is $\Delta_{(\rho,K)}(R)$. No public libraries we found support c-group DP.

Notation

$\mathcal{P}$	Privacy policy for each camera: $\{(\rho, K)_c \;\; \forall \;\; c \;\; \in \;\; \text{cameras}\}$
$\Delta_{\mathcal{P}}(R)$	Maximum number of rows in relation R that could differ by the addition or removal of any (ρ, K)-bounded event.
$\tilde{C_r}(R, a)$	Range constraint: range of attribute a in R
$\tilde{C_s}(R)$	Size constraint: upper bound on total number of rows in R
$\varnothing$	Indicates that a relational operator leaves a constraint unbound. If this constraint is required for the aggregation, it must be bound by a predecessor. If it is not required, it can be left unbound.

Aggregation Functions

Function	**Definition**	**Constraints**	**Sensitivity ($\mathbf{\Delta(Q)}$)**
Count	$Q := \Pi_{\text{count}(*)}(R)$	Δ	$1 \cdot \Delta(R)$
Sum	$Q := \Pi_{\text{sum}(a)}(R)$	$\Delta, \tilde{C_r}$	$\Delta(R) \cdot \tilde{C_r}(R, a)$
Average	$Q := \Pi_{\text{avg}(a)}(R)$	$\Delta, \tilde{C_r}, \tilde{C_s}$	$\frac{\Delta(R) \cdot \tilde{C_r}(R,a)}{\tilde{C_s}(R)}$
Std. Dev	$Q := \Pi_{\text{stddev}(a)}(R)$	$\Delta, \tilde{C_r}, \tilde{C_s}$	$\Delta(R) \cdot \tilde{C_r}(R, a) / \sqrt{\tilde{C_s}(R)}$
Argmax	$Q := \Pi_{\text{argmax}(a)}(R)$	$\Delta, a \in K$	$\max_{k \in K} \Delta(\sigma_{a=k}(R))$

Relational Operators

Operator	**Type**	**Definition**	$\mathbf{\Delta_{\mathcal{P}}(R')}$	$\mathbf{\tilde{C_r}(R', a_i)}$	$\mathbf{\tilde{C_s}(R')}$
Base Case	Base Table	R	$mr \cdot K \cdot (1 + \lceil \frac{\rho}{c} \rceil)$	$\varnothing$	$\varnothing$
Selection (σ)	Standard: rows from R that match WHERE clause	$R' := \sigma_{\text{WHERE}(\ldots)}(R)$	$\Delta_{\mathcal{P}}(R)$	$\tilde{C_r}(R, a_i)$	$\tilde{C_s}(R)$
	Limit: first x rows from R	$R' := \sigma_{\text{LIMIT}=x}(R)$	$\Delta_{\mathcal{P}}(R)$	$\tilde{C_r}(R, a_i)$	$\min(x, \tilde{C_s}(R))$
Projection (Π)	Standard projection: select attributes $a_i, \ldots$ from R	$R' := \Pi_{a_i, \ldots}$	$\Delta_{\mathcal{P}}(R)$	$\tilde{C_r}(R, a_i)$	$\tilde{C_s}(R)$
	Apply (user-provided, but stateless) f to column a_i	$R' := \Pi_{f(a_i), \ldots}$	$\Delta_{\mathcal{P}}(R)$	$\varnothing$	$\tilde{C_s}(R)$
	Add range constraint to column a_i	$R' := \Pi_{a_i \in [l_i, u_i], \ldots}$	$\Delta_{\mathcal{P}}(R)$	$[l_i, u_i]$ if $a_i \neq \varnothing$ $\tilde{C_r}(R, a_i)$ otherwise	$\tilde{C_s}(R)$
GroupBy (γ)	Group attribute(s) (g_i) are `chunk` or `region`	$R' := {}_{g_j, \ldots}\gamma_{\text{agg}(a_i), \ldots}$ $g_j := \text{chunk} \vert \text{bin(chunk)}$	Equation 4.2	$\Delta(\text{agg}(a_i))$	$\frac{\tilde{C_s}(R)}{\text{(bin size)}}$
	Group attribute(s) (g_j) are *not* `chunk` or `region`	$R' := {}_{g_j, \ldots}\gamma_{\text{agg}(a_i), \ldots}$	$\Delta_{\mathcal{P}}(R)$	$\varnothing$	$\varnothing$
	... discrete set of keys provided for each group	$R' := {}_{g_j \in K_j, \ldots}\gamma_{\text{agg}(a_i), \ldots}$	...	...	$\Pi_j \lvert K_j \rvert$
	... aggregation constrains range: $agg(a_i) \in [l_i, u_i]$	$R' := {}_{g_j, \ldots}\gamma_{\text{agg}(a_i) \in [l_i, u_i], \ldots}$	...	$[l_i, u_i]$ if $a_i \neq \varnothing$ $\tilde{C_r}(R, a_i)$ otherwise	...
Joins* ($\bowtie$)	**immediately* preceeded by GroupBy *over same key(s)* ... equijoin on g_j (intersection on g_j) ... outer join on g_j (union on g_j)	$R' := {}_g\gamma_{\text{agg}(a)}(R_1 \bowtie_g \ldots \bowtie_g R_n)$ $R' := {}_g\gamma_{\text{agg}(a)}(R_1 ⟗_g \ldots ⟗_g R_n)$	$\sum_{i=1}^{n} \Delta_{\mathcal{P}}(R_i)$	(GroupBy rules)	(GroupBy rules)

Figure 4-3: Full set of rules for Privid's sensitivity calculation, defined over a query expressed in relational algebra. All queries contain an aggregation statement at the outer-most layer. The sensitivity of this aggregation is a function of the inner relation R, which itself is a function of any inner relations that compose it. This can be followed all the way down to the base table at the leaf, which ends the recursion, and the sensitivity propagates back to the aggregation at the top. In the same way, range and size constraints propagate as well. The range and size must be constrained somewhere along the way from the base table to the aggregation layer, otherwise the sensitivity is undefined and the query must be rejected.

Lemma 4.3.1. Given a relation R, the rules in Figure 4-3 are an upper bound on the global sensitivity of a (ρ, K)-bounded event in an intermediate table t.

Proof. Proof by induction on the structure of the query.

Case: t. $\Delta_{\mathcal{P}}(t)$ is given directly by Equation 4.2.

Case: $R' := \sigma_\theta(R)$. A selection may remove some rows from R, but it does not add any, or modify any existing ones, so in the worst case an individual can be in just as many rows in R' as in R and thus $\Delta_{\mathcal{P}}(R') \leq \Delta_{\mathcal{P}}(R)$ and the constraints remain the same. If θ includes a LIMIT $= x$ condition, then R' will contain at most x rows, regardless of the number of rows in R.

Case: $R' := \Pi_{a,\ldots}(R)$. A projection never changes the number of rows, nor does it allow the data in one row to influence another row, so in the worst case an individual can be in at most the same number of rows in R' as in R ($\Delta_{\mathcal{P}}(R') \leq \Delta_{\mathcal{P}}(R)$) and the size constraint $\tilde{C}_s(R)$ remains the same. If the projection transforms an attribute by applying a stateless function f to it, then we can no longer many assumptions about the range of values in a ($\tilde{C}_r(R', a) = \varnothing$), but nothing else changes because the stateless nature of the function ensures that data in row cannot influence any others.

Case: GroupBy. A `GROUP BY` over a fixed set of a n keys is equivalent to n separate queries that use the same aggregation function over a $\sigma_{\text{WHERE}\mathit{col}=\mathit{key}}(R)$. If the column being grouped is a user-defined column, Privid requires that the analyst provide the keys directly. If the column being grouped is one of the two implicit columns (chunk or region), then the set of keys is not dependent on the contents of the data (only its length) and thus are fixed regardless.

Case: Join. Consider a query that computes the size of the intersection between two cameras, `PROCESS`'d into intermediate tables t_1 and t_2 respectively. If $\Delta(t_1) = x$ and $\Delta(t_2) = y$, it is tempting to assume $\Delta(t_1 \cap t_2) = \min(x, y)$, because a value needs to appear in both t_1 and t_2 to appear in the intersection. However, because the analyst's executable can populate the table arbitrarily, they can "prime" t_1 with values that would only appear in t_2, and vice versa. As a result, a value need only appear in either t_1 or t_2 to show up in the intersection, and thus $\Delta(t_1 \cap t_2) = x + y$ (the sum of the sensitivities of the tables). □

Theorem 4.3.2. Privid (Algorithm 1) provides (ρ, K)-duration privacy for a query $Q := Agg(R)$ over video V, where R is a series of relational operators applied to V.

4.3.1 Relative Privacy Guarantees

In this section, we prove the Privid satisfies the relative privacy guarantee described in §3.4.1. We begin with a lemma that will be helpful for the proof.

Lemma 4.3.3. Consider an individual x whose appearance is bound by $(\hat{\rho}, \hat{K})$ in front of a camera whose policy is (ρ, K, ϵ). For every query Q there exists $\alpha, \beta \in \mathbb{R}$ such that $\alpha K(1 + \beta\rho) \leq \Delta_{(\rho,K)}(Q) \leq \alpha K(2 + \beta\rho)$.

Proof. Any Privid query must contain some aggregation agg as the outer-most relation, and thus we can write $Q := \Pi_{agg}(R)$. $\Delta_{(\rho,K)}(Q)$ is defined in Figure 4-3 for five possible aggregation operators, which are each a function of $\Delta_{(\rho,K)}(R)$ (the sensitivity of their inner relation R).

First, we will prove these bounds are true for the inner relation $\Delta_{(\rho,K)}(R)$ by induction on R (all rules for $\Delta_{(\rho,K)}(R)$ given by Figure 4-3):

Case (Base): $R := t$ When R is an intermediate Privid table t, its sensitivity is given directly by Equation 4.2, where $\alpha = \mathsf{maxrows}_t$ and $\beta = 1/c$. Note, the $(1 + \cdots)$ and $(2 + \cdots)$ in the lemma inequalities bound $\lceil \frac{\rho}{c} \rceil$.

Case (Selection): $R := \sigma(R')$. When R is a selection from R', $\Delta_{(\rho,K)}(R) = \Delta_{(\rho,K)}(R')$. If $\Delta_{(\rho,K)}(R')$ is bound by the inequalities in the lemma statement, then $\Delta_{(\rho,K)}(R)$ is too.

Case (Projection): $R := \Pi(R')$. Same as selection.

Case (GroupBy and Join): $R := \gamma(R_1 ⟗ \ldots ⟗ R_i)$ When R is a `join` of relations R_i proceeded by a `GroupBy`, $\Delta_{(\rho,K)}(R) = \sum_{i=1}^{N} \Delta_{(\rho,K)}(R_i)$. Let $\Delta_{(\rho,K)}R_i$ be parameterized by α_i and β_i. If each of $\Delta_{(\rho,K)}(R_i)$ are bound by the inequalities in the lemma, then $\sum_i \Delta_{(\rho,K)}(R_i)$ is as well, but with $\alpha = \sum_{i=1}^{N} \alpha_i$ and $\beta = \sum_{i=1}^{N} \beta_i$.

Finally, each of the supported aggregation operators only involve multiplying $\Delta_{(\rho,K)}(R)$ by constants (with respect to ρ and K), and thus these constants can be subsumed into α. $\square$

Theorem 4.3.4. Consider a camera with a fixed policy (ρ, K, ϵ). If an individual x's appearance in front of the camera is bound by some $(\hat{\rho}, \hat{K})$, then Privid (Algorithm 1) effectively protects x with $\hat{\epsilon}$-DP, where $\hat{\epsilon}$ is $O(\frac{\hat{\rho}\hat{K}}{\rho K})\epsilon$, which grows (degrades) as $(\hat{\rho}, \hat{K})$ increase while (ρ, K, ϵ) are fixed, and the constants do not depend on the query.

Proof. Privid uses the Laplace mechanism: it returns $Q(V) + \eta$ to the analyst, where $Q(V)$ is the raw query result, and $\eta \sim \text{Laplace}(0, b)$, $b = \frac{\Delta_{(\rho,K)}(Q)}{\epsilon}$ and $\Delta_{(\rho,K)}(Q)$ is the global sensitivity of the query over any (ρ, K)-neighboring videos. Note that the sensitivity is purely a function of the query, and thus Privid samples noise using the same scale b regardless of how long any individual is actually visible in the video.

By Theorem 4.4.1, this mechanism provides ϵ-DP for all (ρ, K)-bounded events. If we rearrange the equation for b so that $\epsilon = \frac{\Delta_{(\rho,K)}(Q)}{b}$, we can equivalently say that Privid guarantees $\frac{\Delta_{(\rho,K)}(Q)}{b}$-DP for all (ρ, K)-bounded events. Or, more generally, that a particular instantiation of Privid with policy $p = (\rho, K, \epsilon)$ guarantees $\hat{\epsilon}$-DP for all $(\hat{\rho}, \hat{K})$-bounded events in query Q, where [2]

$$\hat{\epsilon}_p(\hat{\rho}, \hat{K}, Q) = \frac{\Delta_{(\hat{\rho},\hat{K})}(Q)}{b} = \frac{\Delta_{(\hat{\rho},\hat{K})}(Q)}{\Delta_{(\rho,K)}(Q)/\epsilon} = \frac{\Delta_{(\hat{\rho},\hat{K})}(Q)}{\Delta_{(\rho,K)}(Q)}\epsilon$$

In other words, for a fixed policy, $\hat{\epsilon}$ defines the effective level of protection provided to an event as a function of the *event's* (not policy's) $(\hat{\rho}, \hat{K})$ bound.

From Lemma 4.3.3, we can bound $\hat{\epsilon}$ as $\frac{\alpha\hat{K}(1+\beta\hat{\rho})}{\alpha K(2+\beta\rho)}\epsilon \le \hat{\epsilon} \le \frac{\alpha\hat{K}(2+\beta\hat{\rho})}{\alpha K(1+\beta\rho)}\epsilon$. To see where this comes from, note that $\hat{\epsilon}$ is minimized when the numerator is minimized (the lower bound from Lemma 4.3.3) and the denominator is maximized (the upper bound from Lemma 4.3.3). The same logic applies to the upper bound on $\hat{\epsilon}$.

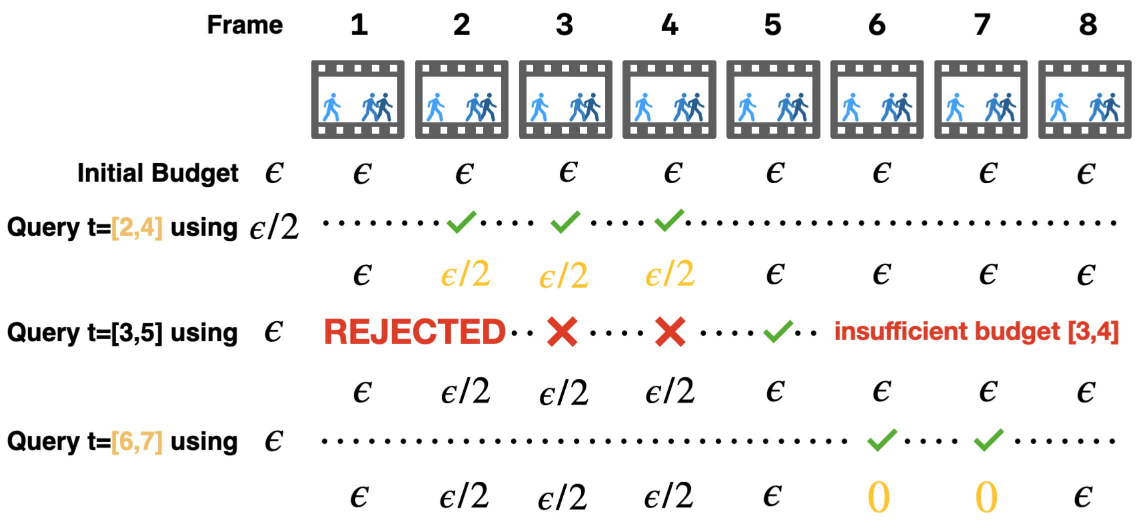
We can simplify both bounds by first canceling α and then picking units of time such that $\beta = 1$ (β has dimensions of chunks per unit time). Thus, $\hat{\epsilon} \approx \frac{\hat{\rho}\hat{K}}{\rho K}\epsilon$. $\square$

4.4 Query Composition

In traditional DP, the parameter ϵ is viewed as a "privacy budget". Informally, ϵ defines the total amount of information that may be released about a database, and each query consumes a portion of this budget. Once the budget is depleted, no further queries can be answered. This ensures that, in the worst case, if all analysts were malicious and coluded, combining their query answers together would still at best satisfy ϵ-DP (not anything weaker).

Since tables are dynamically generated in Privid as an intermediate representation of the underlying video data, we cannot assign a privacy budget to those tables directly, otherwise each query could create a new table based on the same segment of video without depleting the budget. Instead, we allocate budget independently to each *frame* of each camera. The Video Owner can choose to set ϵ however they wish; it may be different between separate cameras or even separate time ranges of the same camera. Each table is tied to the set of frames that generated it. Whenever an ag-

[2]Note the difference in subscript in the numerator and denominator.



Frame
1
2
3
4
5
6
7
8
Initial Budget ϵ
Query t=[2,4] using $\epsilon/2$
$\epsilon/2$
Query t=[3,5] using ϵ
REJECTED
insufficient budget [3,4]
Query t=[6,7] using ϵ
0

Figure 4-4: Example of query budget decrement over a series of three queries. All frames are initialized with ϵ budget. Time flows from top to bottom, where the three queries are executed one after another. We show the updated budget per frame after each query is executed (or not). Query 2 is rejected due to the budget used by Query 1, and thus does not decrement any budget.

gregation and data release is performed on a table, the budget for the corresponding frames are decremented accordingly.

When PRIVID receives a query Q over frames $[a, b]$ requesting budget ϵ_Q, it only accepts the query if *all* frames in the interval $[a-\rho, b+\rho]$ have sufficient budget $\geq \epsilon_Q$, otherwise the query is denied (Alg. 1 Lines 2-4). If the query is accepted, PRIVID then subtracts ϵ_Q from each frame in $[a, b]$, but *not* the ρ margin (Alg. 1 Lines 6-7). We require sufficient budget at the ρ margin to ensure that any single segment of an event (which has duration at most ρ) cannot span two temporally disjoint queries.

Note that since each `SELECT` in a query represents a separate data release, the total budget ϵ_Q used by a query is the sum of the ϵ_i used by each of the i `SELECT`s.

The amount of privacy budget remaining per frame is public information because it is only a function of past queries, not of the data or any query results. If analysts wish to query the interval $[a, b]$ but only part of the range has sufficient budget, they can adjust their interval to only consider frames with sufficient budget.

Example. To put all of this together, we provide an example scenario in Figure 4-4. The video is composed of 8 frames and each is initialized with a budget of ϵ, and ρ is 1 frame. The first query requests to use $\frac{\epsilon}{2}$ over the time (frame) range [2,4]. Since there is enough budget within the ρ margin, i.e. [1,5], the query is accepted and executed, and we decrement $\frac{\epsilon}{2}$ from the range [2,4]. The second query requests ϵ over the time range [3,5]. Thus, we check the range [2,6]. Although t=5 and t=6 have sufficient budget, t=[2,4] does not because of the previous query. Thus, the second query is rejected, no results are released, and no budget is decremented. Since the budget information is public, the analyst could check this for themselves before submitting the query, rather than submitting queries to poll for available budget. Finally, the third query is accepted because we did not lose any budget from the second query. However, t=[6,7] is now completely depleted, so no further queries will be able to analyze them. Further, any queries which include t=6 or t=7 at their margin will also be denied.

Theorem 4.4.1. Consider an adaptive sequence of n queries $Q_1, \ldots, Q_n$, each over the same camera C, a privacy policy (ρ_C, K_C), and global budget ϵ_C. The Privid mechanism (Algorithm 1) provides $(\rho_C, K_C, \epsilon_C)$-privacy for the set of $Q_1, \ldots, Q_n$ in aggregate.

Proof. Consider two queries Q_1 (over time interval I_1, using chunk size c_1 and budget ϵ_1) and Q_2 (over I_2, using c_2 and ϵ_2). Let $v_1 = V[I_1]$ be the segment of video Q_1 analyzes and $v_2 = V[I_2]$ for Q_2. Let E be a (ρ, K)-bounded event.

We say two intervals I_1 and I_2 are "ρ-disjoint" (with $I_1 < I_2$ by symmetry) if the last time unit of I_1 is ρ time units ahead of the first time unit of I_2.

Case 1: I_1 and I_2 are not ρ-disjoint The budget check (lines 1-3 in Algorithm 1) ensures that these two queries must draw from the same privacy budget, because their effective ranges overlap by at least one frame (but may overlap up to all frames). By Theorem 4.3.2, Privid is (ρ, K, ϵ_1)-private for Q_1 and (ρ, K, ϵ_2)-private for Q_2. By Dwork [33, Theorem 3.14], the combination of Q_1 and Q_2 is $(\rho, K, \epsilon_1 + \epsilon_2)$-private.

Case 2: I_1 and I_2 are ρ-disjoint In other words, $I_1 + \rho < I_2 - \rho$, thus the budget check (lines 1-3) allows these two queries to draw from entirely separate privacy budgets. Since the intervals are ρ-disjoint, and all segments in E must have duration $\leq \rho$, it is not possible for the same segment to appear in even a single frame of *both* intervals.

Let K_1 be the number of segments contained in I_1, each of duration $\leq \rho$, and K_2 be the remaining segments contained in I_2, each of duration $\leq \rho$. In other words, E is (ρ, K_1)-bounded in v_1 and (ρ, K_2)-bounded in v_2. Since E has at most K segments, $K_1 + K_2 \leq K$. We need to show that the probability of observing both A_1 and

A_2 if the inputs are the actual segments v_1 and v_2 is close (e^ϵ) to the probability of observing those values if the inputs are the neighboring segments v_1' and v_2':

$$\frac{\Pr[A_1 = Q_1(v_1), A_2 = Q_2(v_2)]}{\Pr[A_1 = Q_1(v_1'), A_2 = Q_2(v_2')]} \leq \exp(e)$$

Since the probability of observing A_1 is independent of observing A_2 (randomness is purely over the noise added by Privid):

$$\begin{aligned}
&\frac{\Pr[A_1 = Q_1(v_1), A_2 = Q_2(v_2)]}{\Pr[A_1 = Q_1(v_1'), A_2 = Q_2(v_2')]} \\
&\leq \frac{\Pr[A_1 = Q_1(v_1)] \Pr[A_2 = Q_2(v_2)]}{\Pr[A_1 = Q_1(v_1')] \Pr[A_2 = Q_2(v_2')]} \\
&\leq \frac{\frac{1}{2b_1}\exp(-\frac{|A_1-Q_1(v_1)|}{b_1})\frac{1}{2b_2}\exp(-\frac{|A_2-Q_2(v_2)|}{b_2})}{\frac{1}{2b_1}\exp(-\frac{|A_1-Q_1(v_1')|}{b_1})\frac{1}{2b_2}\exp(-\frac{|A_2-Q_2(v_2')|}{b_2})} \\
&\quad \text{(By Algorithm 1, Line 13)} \\
&= \exp(\tfrac{|A_1-Q_1(v_1')|-|A_1-Q_1(v_1)|}{b_1} + \tfrac{|A_2-Q_2(v_2')|-|A_2-Q_2(v_2)|}{b_2})
\end{aligned}$$

If K_1 segments are in v_1 and K_2 segments are in v_2, the numerator of each fraction above is the sensitivity of a (ρ, K_1)-bounded event and a (ρ, K_2)-bounded event, respectively. b_1 and b_2 are the amount of noise actually added to the query, which are both based on K:

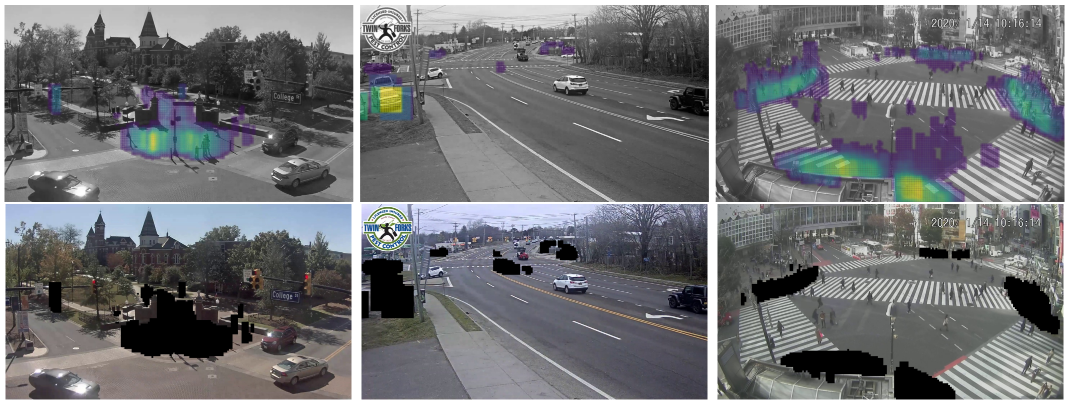
$$\begin{aligned}
&\leq \exp(\frac{\Delta_{(\rho,K_1)}(Q_1)}{\Delta_{(\rho,K)}(Q_1)/\epsilon} + \frac{\Delta_{(\rho,K_2)}(Q_2)}{\Delta_{(\rho,K)}(Q_2)/\epsilon}) \\
&= \exp(\epsilon \cdot (\frac{K_1(\lceil \frac{\rho}{c_1} \rceil + 1)}{K(\lceil \frac{\rho}{c_1} \rceil + 1)} + \frac{K_2(\lceil \frac{\rho}{c_2} \rceil + 1)}{K(\lceil \frac{\rho}{c_2} \rceil + 1)})) \\
&\quad \text{(by Equation 4.2)} \\
&= \exp(\epsilon \cdot (\frac{K_1}{K} + \frac{K_2}{K})) \quad \text{(recall } K \geq K_1 + K_2) \\
&\leq \exp(\epsilon)
\end{aligned}$$

□

4.5 Spatial Masking

4.5.1 Intuition

Observation. In some settings, a few individuals may be visible to a camera for far longer than the majority, creating a heavy-tailed distribution of presence durations.



(a) `campus` (b) `highway` (c) `urban`

Figure 4-5: (Top) Heatmap measuring the maximum time any object spent in each pixel, noramlized to the max (yellow) per video. (Bottom) The resulting masks used for our evaluation, chosen from the list of masks automatically generated using Algorithm 2.

For example, on a city street, most cars are visible for a short time as they drive past the camera, while a small number of cars may be visible for hours if they park on the street in front of the camera. Alternatively, most individuals are visible briefly as they walk down a sidewalk, while a few are visible for far longer if they stop to sit at a bench or table. The maximum duration in such distributions is tight only for those few lingering individuals, and is a "loose" bound for everyone else. Setting (ρ, K) to this maximum will result in a large amount of noise to protect those individuals at the tail; all others could have been protected with a far lower amount of noise.

As in the motivating examples above, we observe that these "lingering" individuals at the tail tend to spend the majority of their time in one of a few fixed regions in the scene, but a relatively short time in the rest of the scene. For example, a car may be parked in a parking spot for hours, but only visible for 1 minute while entering/leaving the spot. Further, these regions tend to be static over time and are thus amenable to discovery by analysis of past data from a camera.

We demonstrate this observation for 3 videos from our evaluation in the top row of Figure 4-5. For each scene, we analyzed 12 hours of video data and computed a "duration heatmap" visualizing the amount of time individuals (people or cars) occupied each region of the frame relative to the maximum duration object. Yellow denotes regions where individuals spent the most time, while grey denotes regions where individuals spent a very short time relative to yellow. In `campus`, the colored region in the center corresponds to a street corner where individuals wait for a few minutes to cross the street during a red light. In `highway`, the colored regions in the top left and right cover spots where cars are parked for hours at a time. In `urban`, the colored regions again represent street corners where individuals wait to cross.

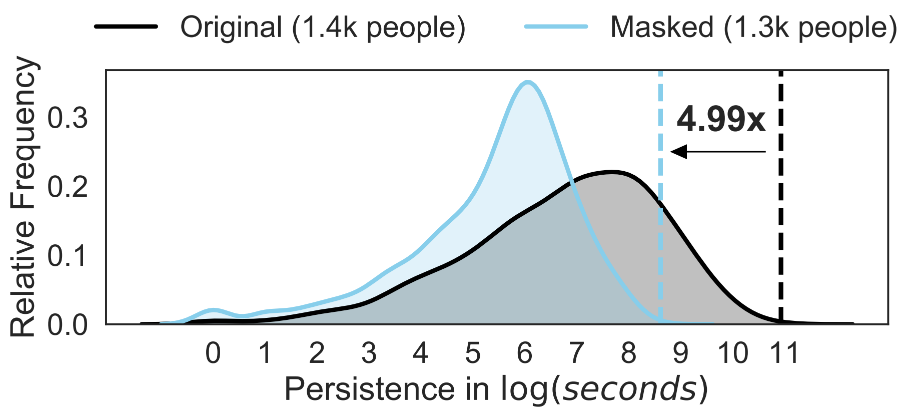
Opportunity. Masking fixed regions (i.e., removing those pixels from all frames prior to running the analyst's video processing) in the scene that contain lingering individuals can drastically reduce the *observable* maximum duration of those individuals. For example, the parked car mentioned in the previous paragraph would be observable for 1 minute rather than hours. This would permit a tighter policy (i.e., a lower value of ρ) that provides the same *semantic* guarantee–all appearances would still be bound by the policy. To demonstrate this opportunity, we use the heatmaps in the top row of Figure 4-5 to generate the masks in the bottom row of Figure 4-5. In Figure 4-6 we plot the distribution of durations before and after applying the mask. In each case, the mask significantly reduces the maximum duration, resulting in a factor of 1.71-9.65$\times$ less noise, while blocking only a relatively small fraction of the target objects.

Of course, this technique is only useful to an analyst if the remaining (unmasked) part of the scene includes all the information needed for the query at hand. For example, if counting cars, masking sidewalks would be reasonable, but masking roads would not.

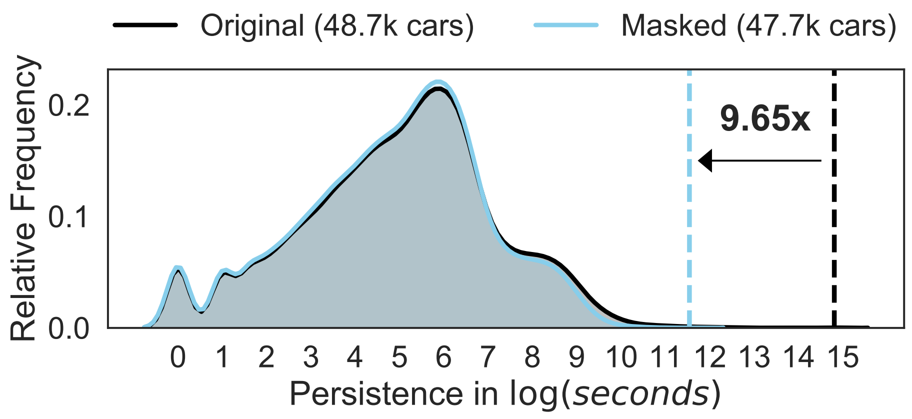
4.5.2 Usage in Privid

At camera-registration time, instead of providing a single (ρ, K) policy to Privid (for a given camera), the Video Owner can provide a (fixed) "menu" of a few frame masks and, for each, a corresponding (ρ, K) policy that would provide a semantically-equivalent level of privacy when that mask is applied. At query time, the analyst can choose a mask from the menu that would minimally impact their query goal while maximizing the reduction in noise (via the tighter (ρ, K) bound). They specify a mask using the optional `WITH_MASK [NAME]` directive in a `split_stmt` (or simply leave out the directive for no mask). Privid applies the mask to all video frames in that video block *before* passing it to the analyst's `PROCESS` executable (i.e., the analyst's executable only "sees" the masked video), and uses the corresponding (ρ, K) in its sensitivity calculation (§4.3).

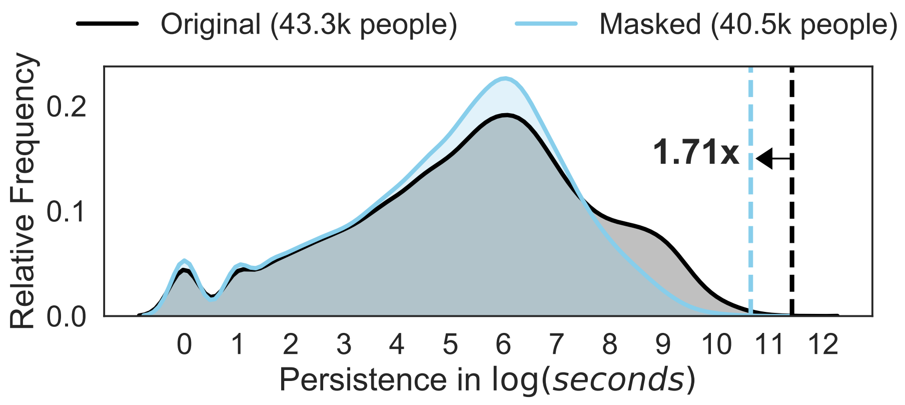
It is the Video Owner's responsibility to choose the menu of masks and policies and to ensure that those policies match the definition of privacy they desire. For example, if the Video Owner came up with an initial bound on ρ by analyzing prior video from the camera, they can apply that same process (but with the mask applied to the



(a) campus



(b) highway



(c) urban

Figure 4-6: The distribution of private objects' durations (persistence) is heavy-tailed. Applying the masks from the bottom row of Figure 4-5 significantly lowers the maximum duration, while still allowing most target objects to be detected. The key denotes the total number of target objects detectable before and after applying the mask. The dotted lines highlight the maximum persistence, and the arrow text denotes the relative reduction in maximum persistence after applying the mask.

video first) to estimate the maximum *observable* duration of objects they wish to protect. PRIVID takes a menu of masks and policies as input and simply ensures that the correct policy is enforced. It does not make any guarantees about the semantic meaning of a policy.

Regardless of how the masks are chosen, the masks themselves are static (i.e., the same pixels are masked in every frame regardless of its contents), and the set of available masks is fixed. Neither depend on the query itself or the specific content of the video being queried. Further, the mask itself does not reveal anything about how the Video Owner generated it or which specific objects contributed to it, it only tells the analyst that some objects appear for a long duration in the masked region.

4.5.3 Choosing Masks

Which masks should the Video Owner provide? The space of all possible masks ($O(2^{1920\times1080})$ for 1080p video) is infeasible to enumerate, and even computing the ρ bound for a single iteration (via object tracking) is computationally expensive. However, the set of masks that would actually reduce the observable duration is relatively small and query-agnostic, so exploring the entire space is unnecssary. For example, suppose we create a mask that removes only the roads in `campus`. Since the maximum duration individuals in this scene are present on the sidewalks, this mask would not actually reduce the observable ρ at all.

Recall that our goal in choosing a masked region is to *reduce* ρ. To do this, note that ρ is defined by the maximum duration individual. In order to decrease ρ, we must decrease the observable duration of this individual. While masking any pixels this individual visited could decrease ρ, the smallest region that will decrease ρ the most is exactly the highlighted region in our heatmaps from Figure 4-5.

We can use this intuition to create an efficient recursive algorithm which requires only a single run of the (expensive) object tracking routine. We begin with the empty mask, and we will progressively add more pixels to the mask, creating a checkpoint of mask options along the way. First, we compute object tracks for a portion of past video and create a duration heatmap. We can compute an estimate of (ρ, K) based on these object tracks. This provides the initial policy for the null mask. Then, we pick the highest duration region from the duration heatmap, and add this to our running mask. We adjust the object tracks to remove portions that would not have been visible with this mask, and recompute a (ρ, K) policy for this mask, which becomes the next option in the menu. At this point, we proceedrecursively. We consider the

Algorithm 2: Generate Menu of Useful Masks

```
  ▷ Input  : $n$,$m$: camera resolution ($n \times m$)
  ▷ Input  : $g$ : grid size, frame pixels will be divided evenly into boxes of size
             $b \times b$
  ▷ Input  : $objs$: list of all object tracks, each object track is a list of
             (frame_id, bounding_box) tuples corresponding to the same
             object
  ▷ Output: $menu$: list of tuples of $(mask, policy)$, mask is an $n \times m$ binary
             matrix, policy is a $(\rho, K)$ bound for the corresponding mask
1 $obj\_bounds \leftarrow$ array of tightest $(\rho, K)$ bound for each track in $objs$
  ▷ Initialize the empty mask and corresponding policy
2 $curr\_mask \leftarrow$ []
3 $curr\_policy \leftarrow max(obj\_bounds)$
4 $last\_policy \leftarrow curr\_policy$
5 $menu = [(curr\_mask, curr\_policy)]$
  ▷ Add boxes to the mask one by one
6 $B \leftarrow$ set of pixel boxes available to be masked, $n \times m$ divided into $g \times g$ boxes
7 while $B$ is not empty do
      ▷ Pick the box that, if removed, will reduce the bound the most
8     $max\_obj = o \in obj\_bounds$ with largest $(\rho, K)$ bound
9     $max\_grid\_box = b \in B$ that overlaps $max\_obj$ for max number of
        frames
10    $curr\_mask += max\_grid\_box$
      ▷ Remove any detections that are no longer observable after applying the
        mask and recompute the new tightest observable policy
11    $objs -= $ any bounding boxes that are completely covered by $curr\_mask$
12    $obj\_bounds \leftarrow$ recompute tighetst $(\rho, K)$ bounds for each track in $objs$
13    $curr\_policy \leftarrow max(obj\_bounds)$
      ▷ Add this mask to the menu if this policy improves on the bound
14    if $curr\_policy$ tighter than $last\_policy$ then
15        $menu += (curr\_mask, curr\_policy)$
16    $last\_policy \leftarrow curr\_policy$
17    $B -= max\_grid\_box$
```

highest duration unmasked region, add it to the mask, recompute the policy, then add the current mask and policy as another option in the menu. Eventually, either we will reach a policy of $\rho = 0$, or the entire mask will be added to the frame, at which point we are done. We provide this algorithm as pseudocode in Algorithm 2.

It is important to note here that adding a masked region may create a discontinuity in an object's appearance and thus require increasing K. Consider the example car from earlier. While they were originally bound by $(\rho = 61min, K = 1)$, adding a mask around the parking spot could change their observable bound to $(\rho = 45sec, K = 2)$ (suppose they are visible for 45 seconds while entering the spot and then 15 seconds at a different time while leaving the spot).

For a visual intuition of this algorithm, imagine the heatmaps in Figure 4-5 denote a convex surface where the color denotes the depth, with grey being the top level and yellow being the deepest point. We can think of this procedure as a water-filling algorithm, where adding more water to the surface is equivalent to increasing the size of the mask.

4.5.4 Microbenchmarks

We validate the effectiveness of masking over 3 videos from our own dataset, and 7 other videos from the datasets of related work (BlazeIt [45] and Miris [23]). For each video, we run Algorithm 2 to generate a series of masks and policies. Figure 4-7 plots a summary. The x-axis represents the fraction of the frame that has been included in the mask. Initially, we begin with the null mask, so none of the frame is masked $(x = 0)$. As we run the algorithm, we progressively mask a greater fraction of the frame, up until the entire frame is masked at $x = 100$ (%). Each x value represents a particular mask.

In the top plot, the y-axis measures the maximum duration ("persistence") of any object observable with that mask applied, normalized to the persistence with no mask applied (so that we can easily compare videos with different ρ in the same graph). By definition, y will be 100% of $x = 0$ at $x = 0$ and 0% at $x = 100$ when the entire frame is masked and nothing is visible.

In the bottom plot, the y-axis measures the number of objects that remain visible with that mask applied. Again this value is normalized to the number of objects visible with no mask applied.

For each video, we can see that a large drop in persistence occurs *before* a large drop in the number of objects. In other words, for each video, there exists a mask

Before Mask			**After Mask**			
Dataset	**Video Name**	**Max Duration (Frames)**	**%Frame Masked**	**Max Duration (Frames)**	**Duration Reduction**	**% Objects Retained**
Privid	`campus`	1951	17%	190	10.27x	91.06%
	`highway`	28800	30%	601	47.92x	91.3%
	`urban`	2746	19%	497.16	5.52x	87.24%
BlazeIt [45]	grand-canal	10930	35%	2496	4.38x	26.67%
	venice-rialto	37992	6%	7696	4.94x	94.21%
	taipei	56931	20%	2444	23.29x	99.94%
Miris [23]	shibuya	9363	2%	2182	4.29x	96.43%
	beach	4843	5%	843.2	5.74x	94.79%
	warsaw	6479	4%	1147	5.65x	94.82%
	uav	595	40%	130	4.58x	75.57%

Table 4-1: Effectiveness of applying masks in terms of reducing the observable duration with obscuring too many target objects.

that significantly reduces the maximum observable duration without removing too many of the target objects.

To highlight this, we pick a particular mask for each video and display the effectiveness in Table 4-1. The left side of the table displays the max duration ρ before applying any mask. The right side shows the impact of the mask. "% Frame Masked" is equivalent to the x value from Figure 4-7, indicating the fraction of pixels that the mask covers.The remaining columns show the max duration ρ after applying that mask, the relative reduction in ρ, and the fraction of objects that are still visible. Ideally a larger duration reduction and a higher fraction of objects retained are better. In each case, we were able to find a mask that reduces ρ by 4-24x (indicating a proportional increase in query accuracy due to lower noise) while retaining a majority of the objects, ensuring the query still has a representative sample to analyze.

4.5.5 I thought you said denaturing was bad!

Although masking is a form of denaturing, Privid uses it differently than the prior approaches in §2.1, in order to sidestep their issues. Rather than attempting to dynamically hide individuals as they move through the scene, Privid's masks cover a *fixed* location in the scene and are publicly available so analysts can account for them in their query implementation. Also, masks are used as an optional modification to the input video; the rest of the Privid pipeline, and thus its formal privacy guarantees, remain the same.

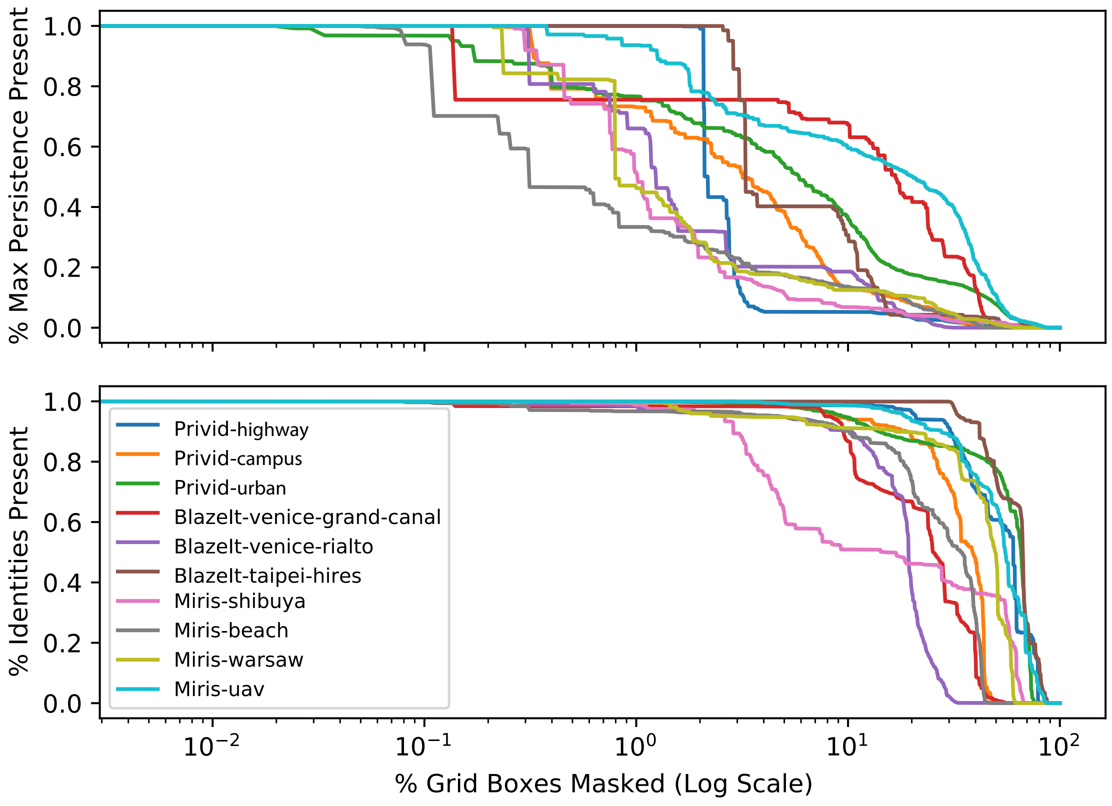


Figure 4-7: Cumulative impact of masking on the maximum observable duration and number of target objects retained. Boxes are masked in the order calculated by Algorithm 2. In both plots, the y-axis is scaled relative to $x = 0$ for ease of visualization. We chose a particular x for each video and show the absolute x and y values in Table 4-1.

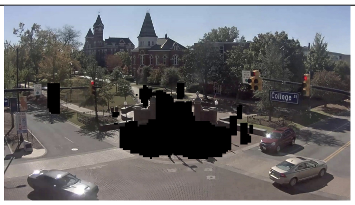
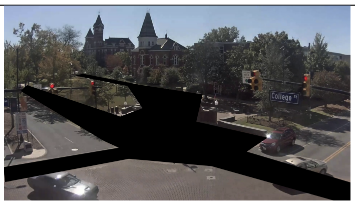
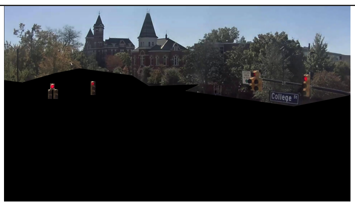
Name	View	Region Masked	Ideal For	Policy
None				$(\rho = 250s, K = 1)$
A		Street corner	People	$(\rho = 24s, K = 2)$
B		All walkways (no people visible)	Cars	$(\rho = 15s, K = 2)$
C		No people or cars visible	Traffic lights, trees, sky	$(\rho = 0s, K = 0)$

Table 4-2: Example of menu of semantically-useful masks that could be generated for the `campus` camera.

Video	Max(frame)	Max(region)	Reduction
`campus`	6	3	2.00×
`highway`	40	23	1.74×
`urban`	37	16	2.25×

Table 4-3: Reduction in max output range from splitting each video into distinct regions. Reduction shows the factor by which the noise could be reduced. A reduction of the max by 2× corresponds to a reduction in the level of noise by 2× as well.

4.6 Spatial Splitting

4.6.1 Intuition

Observation. (1) At any point in time, each object typically occupies a relatively small area of a video frame. (2) Many common queries (e.g., object detections) do not need to examine the entire contents of a frame at once, i.e., if the video is split spatially into regions, they can compute the same total result by processing each of the regions separately.

Opportunity. Privid already splits videos temporally into chunks. If each chunk is further divided into spatial regions and an individual can only appear in one of these chunks at a time, then their presence occupies a relatively smaller portion of the intermediate table (and thus requires less noise to protect). Additionally, the maximum duration of each individual region may be smaller than the frame as a whole.

4.6.2 Usage in Privid

At camera-registration time, Privid allows Video Owners to manually specify boundaries for dividing the scene into regions. They must also specify whether the boundaries are soft (individuals may cross them over time, e.g., between two crosswalks) or hard (individuals will never cross them, e.g., between opposite directions on a highway). At query time, analysts can optionally choose to spatially split the video using these boundaries. Note that this is in addition to, rather than in replacement of, the temporal splitting. If the boundaries are soft, tables created using that split must use a chunk size of 1 frame to ensure that an individual can always be in at most 1 chunk. If the boundaries are hard, there are no restrictions on chunk size since the Video Owner has stated the constraint will always be true.

4.6.3 Microbenchmarks

We demonstrate the potential benefit of spatial splitting on three videos from our evaluation (Q1-Q3). For each video, we manually chose intuitive regions: a separate region for each crosswalk in `campus` and `urban` (2 and 4, respectively), and a separate region for each direction of the road in `highway`. Table 4-3 compares the range necessary to capture all objects that appear within one chunk in the entire frame compared to the individual regions. The difference (1.74-2.25$\times$) represents the potential noise reductions from splitting: noise is proportional to max(frame) or max(region) when splitting is disabled or enabled, respectively.

4.6.4 Discussion

To increase the applicability of spatial splitting, Privid could allow analysts to divide each frame into a grid and remove the restrictions on soft boundaries to allow any chunk size. This would require additional estimates about the max size of any private object (dictating the max number of regions they could occupy at any time), and the maximum speed of any object across the frame (dictating the max number of regions they could move between). We leave this to future work.

4.7 Example Query

In this section, we consider a very simple hypothetical query as an example to tie everything together. We will first show a benevolent query, then show a malicious query that *looks* the same from Privid's perspective, and explain how Privid's privacy protection works without needing to differentiate intent.

4.7.1 Benevolent Query

Suppose a Video Owner provides access to `camA` via Privid, with a policy ($\rho = 60s, K = 2$). The city transportation department wishes to collect statistics about vehicles passing `camA` during October 2021. We formulate two questions as a Privid query:

The `SPLIT` selects 1 month of video from `camA`, then divides the frames into a list of 10-second-long chunks (267k chunks total). The `PROCESS` first creates an empty table based on the `SCHEMA` (3 columns). Then, for each chunk, it starts a fresh instance of `traffic_flow.py` inside a restricted container, provides the chunk as input, and

```
-- Select 1 month time window from camera, split into chunks
SPLIT camA
    BEGIN 10-01-2021/12:00am END 11-01-2021/12:00am
    BY TIME 10sec STRIDE 0sec
    INTO chunksA;
-- Process chunks using analyst's code, store outputs in tableA
PROCESS chunksA USING traffic_flow.py TIMEOUT 1sec
    PRODUCING 20 ROWS
    WITH SCHEMA (plate:STRING="", type:STRING="", speed:NUMBER=0)
    INTO vehiclesA;
-- S1: Number of unique cars per day
SELECT day,COUNT(DISTINCT plate) FROM vehiclesA WHERE type=="car"
        GROUP BY day CONSUMING eps=0.5;
-- S2: Average speed of trucks
SELECT AVG(range(speed, 30, 60)) FROM vehiclesA WHERE type=="truck"
        CONSUMING eps=0.5;
```

Figure 4-8: Example PRIVID query, whose intent (benevolent or malicious) is unknown. The intent depends partially on the `PROCESS` executable (which we do not vet) and partially on the analyst's interpretation of the output (which we cannot anticipate).

```
traffic_flow.py
import detectron
import deepsort
import openalpr

tracker = deepsort.Tracker()

for frame in sys.stdin.buffer.read(FRAME_SIZE_BYTES):
    objects = detectron.detect(frame)
    for car in filter(objects, label="car"):
        plate = openalpr.process(car)
        car.plate = plate
        color = compute_obj_color(car)
        car.color = color
    tracker.add(objects)

for car in tracker:
    print(car.plate, car.color, car.speed)
```

Figure 4-9: `PROCESS` executable referenced by benevolent analyst's query (Figure 4-8)

appends the output as rows to `vehiclesA`. The executable `traffic_flow.py` (Figure 4-9) contains off-the-shelf object detection and tracking models, a license plate reader, and a speed estimation algorithm.

The first `SELECT` filters all cars, then counts the "distinct" license plates to estimate the number of *unique* cars per day. Each day is a separate data release with an independent sample of noise. The second `SELECT` filters all trucks, then computes the average speed across the entire month of footage. It uses the same input video as the first select, and thus draws from the same budget, so in aggregate the two `SELECT`s consume $\epsilon = 1.0$ budget from all frames in October 2021.

4.7.2 Malicious Query Attempt

Now consider a malicious analyst Mallory who wishes to determine if individual x appeared in front of `camA` each day. Assume x's appearance is bound by the Video Owner's (ρ, K) policy.

To hide their intent, Mallory disguises their query as a traffic counter, mimicking S_1 from the previous example. They write identical query statements, but their "`traffic_flow.py`" instead includes specialized models to detect x. If x appears, it outputs 20 rows (the maximum) with random values for each of the columns, otherwise it outputs 0 rows. This adds 20 rows to the corresponding daily count for each chunk x appears.

Amplification attempt. Due to the isolated environment (§4.2.5), the `PROCESS` executable can only output rows for a chunk if x *truly appears.* It has no way of saving state or communicating between executions in order to artificially output rows for a chunk in which x does not appear. It could output more than 20 rows for a single chunk, but Privid ignores any rows beyond the `PROCESS`'s explicit max (20), so this would not increase the count. Increasing the rows per chunk parameter would also be pointless: Privid would compute a proportionally higher sensitivity and add proportionally higher noise.

Side channel attempt. The executable could try to encode the entire contents of a frame in a row of the table, either by encoding it as a string, or a very large number of individual integer columns. But in either case, the analyst cannot view the table directly or even a single row directly, it can only compute noisy aggregations over entire columns.

Summary. Privid would compute the sensitivity of S_1 (identical in both the benevolent and malicious cases) as $\Delta_{(60,2)}(Q) \leq 20 \cdot 2 \cdot (1 + \lceil \frac{60}{10} \rceil) = 280$ rows, meaning it would add noise with scale 280 to each daily count. Regardless of how Mallory changes her executable, it cannot output more than 280 rows based on x's presence. Thus, even if she observed a non-zero value ~ 280, she could not distinguish whether it is a result of the noise or x's appearance.

Mallory's query gets a useless result, because her target (x's appearance) was close in duration to the policy. In contrast, the benevolent query can get a useful result because the duration of its target (the set of *all cars'* appearances) far exceeds the policy. PRIVID's noise will translate to $\mathcal{L}^{-}1(p = 0.99, u = 0, b = \frac{\Delta}{\epsilon} = \frac{280}{0.5}) \leq 2200$ cars with 99% confidence. If, for example, there are an average of 10 cars in each chunk (and thus 86000 in one day), 2200 represents an error of $\pm 2.5\%$.

4.8 Limitations

PRIVID does not explicitly prevent many queries from being "expressed", but it does prevent some classes of queries from achieving reasonable utility. In this sense, PRIVID does not support the following classes of queries:

- Fine-grained queries, such as those explicitly looking for a particular individual.
- Unlimited queries over the same portion of video.
- Processing algorithms that require maintaining state over the whole video.

Chapter 5

Evaluation

In this chapter, we demonstrate that Privid supports a diverse range of video analytics queries, including filtered object counting, duration queries, and multi-camera aggregations over three real-world video streams. For each, we show that Privid can simultaneously achieve both high utility and a meaningful privacy guarantee: we choose a policy for each video stream that protects the presence of *all* individuals, and Privid only increases error by 1-5% relative to a non-private system.

5.1 Setup

Datasets. We evaluated Privid primarily using three representative video streams (`campus`, `highway` and `urban`, screenshots in Figure 4-5) that we collected from YouTube spanning 12 hours each (6am-6pm). These videos cover low and high object density scenarios as well as a mix of people and cars. They are also representative of typical camera resolutions (1080p) and angles.

For the multi-camera case study (Case 2), we use the Porto Taxi dataset [59] containing 1.7mil trajectories of all 442 taxis running in the city of Porto, Portugal from Jan. 2013 to July 2014. We apply the same processing as [40] to emulate a city-wide camera dataset; the result is the set of timestamps each taxi would have been visible to each of 143 cameras over the 1.5 year period.

Implementation. Our prototype implementation of Privid consists of a few separate parts:

- The query sensitivity engine is implemented using 2k lines of Rust. In particular, we define the language as a parsing expression grammar and use the Pest [19] crate (library) to build the query parser.

- The execution environment is described in §4.2.5 and uses standard Linux security primitives: chroot, AppArmor, namespaces, and cgroups.

- The overall query engine which handles actually splitting the videos, running the analyst-provided processing code, creating the intermediate tables, and computing raw (pre-noise) results is implemented in 4k lines of Python.

- For all `PROCESS` executables and camera-owner (ρ, K) estimation, we used the Faster-RCNN [72] model in Detectron-v2 [85] for object detection, and DeepSORT [82] for object tracking. For these models to work reasonably given the diverse content of the videos, we chose the hyperparameters for detection and tracking on a per-video basis (details in §6.1.2).

Privacy policies. We assume the Video Owner's underlying privacy goal is to "protect the appearance of all individuals". For each camera, we use the strategy in §4.5, to create a map between masks and (ρ, K) policies that achieve this goal. All policies use $K = 1$ to protect single appearances since our video data did not indicate reappearances were typical or expected.

Privacy budget. We assign a budget of $\epsilon = 1$ to all frames of each camera as this is well-accepted as a reasonable level of privacy in many standard settings.This translates to protecting the appearance of each individual with $\epsilon = 1$-DP. We consider each case study independently. That is, the full budget is available to the set of queries in each case study, but must be shared among queries in the same case study. If our dataset spanned a longer time frame, we could simply execute them over disjoint ranges of time to achieve this.

Baselines. For each query, we compute `accuracy` by comparing the output of Privid to running the same exact query implementation without Privid. We execute each query 1000 times, and report the mean accuracy value $\pm$ 1 standard deviation.

5.2 Query Case Studies

We formulate five types of queries to span a variety of axes (target object class, number of cameras, aggregation type, query duration, standing vs. one-off query). Figure 5-1 displays hourly results for Q1-Q3 as a time-series. Table 5-1 summarizes the remaining queries (Q4-Q13), which return only a single value (shown in the "Query Output" column).

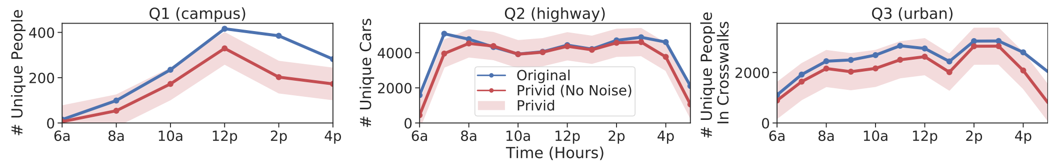


Figure 5-1: Time series of Privid's output for Case 1 queries. "Original" is the baseline query output without using Privid. "Privid (No Noise)" shows the raw output of Privid before noise is added. The final noisy output will fall within the range of the red ribbon 99% of the time.

Case #	**Q#**	**Query Description**	**Query Parameters**	**Video**	ρ	**Query Output**	**Accuracy**
Case 1	Q1	Count Unique People Per Hour	$\lvert W\rvert = 1$ hour, $c = 30$ sec, Agg $=$ `SUM`, range $= (0, 6)$	`campus`	49 sec	Fig. 5-1	90% ± X
Case 1	Q2	Count Unique Cars Per Hour	$c = 2.0$ min, range $= (0, 100)$	`campus`	2.0 min	Fig. 5-1	90% ± X
Case 1	Q3	Count Unique People Per Hour (Filter: in crosswalks)	$c = 30$ sec, range $= (0, 23)$	`campus`	3.3 min	Fig. 5-1	90% ± X
Case 2	Q4	Average Taxi Driver Working Hours (`union` across 2 cameras)	$\lvert W\rvert = 365$ days, $c = 15$ sec, Agg $=$ `avg`, range $= (0, 16)$	`porto10`, `porto27`	[45, 195] sec	5.87 hrs	$94.14\% \pm 0.18\%$
Case 2	Q5	Average # Taxis Traversing 2 Locations on Same Day (`intersection` across 2 cameras)	$\lvert W\rvert = 365$ days, $c = 15$ sec, Agg $=$ `avg`, range $= (0, 300)$	`porto10`, `porto27`	[45, 195] sec	131 taxis	$99.80\% \pm 0.13\%$
Case 2	Q6	Identifying Camera with Highest Daily Traffic (`argmax` across all 143 cameras)	$\lvert W\rvert = 365$ days, $c = 15$ sec, Agg $=$ `argmax`	`porto1`, ..., `porto143`	[15, 525] sec	`porto20`	100.00%
Case 3	Q7	Fraction of trees with leaves (%)	$\lvert W\rvert = 12$ hrs, $c = 1$ frame, Agg $=$ `avg`, range $= (0, 100)$	`campus`	49 sec	$15/15 = 1.00$	$99.90\% \pm 0.11\%$
	Q8			`highway`	6.21 min	$3/7 = 0.43$	$98.24\% \pm 1.90\%$
	Q9			`urban`	3.34 min	$4/6 = 0.67$	$99.39\% \pm 0.66\%$
Case 4	Q10	Duration of Red Light (seconds)	$\lvert W\rvert = 12$ hrs, $c = 10$ min, Agg $=$ `avg`, range $= (0, 300)$	`campus`	0 sec	75 sec	100.00%
	Q11			`highway`	0 sec	50 sec	100.00%
	Q12			`urban`	0 sec	100 sec	100.00%
Case 5	Q13	# Unique People (Filter: trajectory moving towards campus)	$\lvert W\rvert = 12$ hrs, $c = 10$ min, Agg $=$ `sum`, range $= (0, 25)$	`campus`	49 sec	576 people	$79.06\% \pm 4.75\%$

Table 5-1: Summary of query results across all case studies.

5.2.1 Case 1: Counting Private Objects Over Time (Q1-Q3)

One of the most common and useful query primitives for large-scale video analytics systems is a count of unique objects over time. To demonstate Privid's support for standing queries and short (1 hour) aggregation durations, we compute a count of unique objects observed *per hour* over the 12 hours in each of our videos. For each query, we first describe our design decisions, then provide the PQL statement we constructed as a result.

Q1. Our goal is to count the number of unique people observed per hour in `campus`. One consideration is whether or not to use a mask. A menu of possible masks for `campus` is shown in Table 4-2. Since we only need to observe someone once to count them, we can use a mask that removes the center portion ("A"): all individuals walk from one edge of the frame to the other and will be countable before and after they enter this region. Any masks that remove more of the frame would disrupt our count and may miss some individuals.

To ensure that we do not double count individuals who either appear in multiple chunks or who pass through the masked region during a single chunk, our process executable, `count_ppl_campus.py` only counts individuals that *enter* the frame within a chunk. Once an individual enters, they are ignored. Thus, whether or not they pass through the masked region does not impact any future counts. We do not need to maintain any information about each person, so we create a table with a single column `ppl` and output a single row per chunk that corresponds to the number of people observed during that chunk.

Finally, we construct our aggregation. All Privid tables include a system-provided `chunk` column which can be used to aggregate over time. To release one result per hour, we use a GROUP BY over the chunk column, with the `hour(chunk)` helper to transform each chunk to the corresponding hour. For each hour, we want to sum the `ppl` column. However, since the `ppl` column does not have a range constraint by default, we must supply one. To choose this value, we manually observe the sample clip. This intersection is not very busy, and the clip indicates that there are typically at most 6 people entering at any one time. If there are any chunks where more than 6 people enter, our result will be trimmed to 6. However, as long as this is not persistent, this is likely a favorable tradeoff: increasing the range would incur more noise.

Case 1: Query 1

```
SPLIT campusCam
    BEGIN 06-01-2019/06:00am END 06-01-2019/06:00pm
    BY TIME 30sec STRIDE 0sec
    WITH MASK C1
    INTO campusChunks;
PROCESS campusChunks USING count_ppl_campus.py TIMEOUT 1sec
    PRODUCING 1 ROWS
    WITH SCHEMA (ppl:NUMBER=0)
    INTO campusTable;
SELECT hour,sum(RANGE(ppl,0,6)) from campusTable
    GROUP BY hour(chunk)
    CONSUMING eps=1.0;
```

Q2. Our goal is to count the number of unique *cars* per hour in `highway`. Most of our decisions here are analogous to Q1. We choose a mask that removes parked cars, but leaves the highway portion visible. Similarly, our process executable just outputs the number of cars per chunk. Our executable, `count_cars.py` addresses double counting by only counting cars that pass a virtual line in the road, rather than all cars visible in the frame. The sample clip indicates that this portion of road can get quite busy, we estimate that there are typically at most 100 cars in any 2 minute period, which gives our range constraint.

Case 1: Query 2

```
SPLIT highwayCam
    BEGIN 06-01-2019/06:00am END 06-01-2019/06:00pm
    BY TIME 2min STRIDE 0sec
    WITH MASK H2
    INTO highwayChunks;
PROCESS highwayChunks USING count_cars.py TIMEOUT 1sec
    PRODUCING 1 ROWS
    WITH SCHEMA (cars:NUMBER=0)
    INTO highwayTable;
SELECT hour, sum(RANGE(cars,0,100)) from highwayTable
    GROUP BY hour
    CONSUMING eps=1.0;
```

Q3. Our goal is to count the unique people crossing the intersection per hour in `urban`. Since some individuals loiter in other parts of the frame, we choose a mask that removes all pixels except the crosswalks to get the tighest possible policy (Figure 4-5c). This scenario is also a good candidate for spatial splitting (§4.6): an individual can only be in one crosswalk region at a time, and the number of people is fairly evenly distributed across crosswalks. We specify this using `BY REGION ‘‘crosswalk’’`.

While there are typically at most 40 people visible across all of the crosswalks, there are at most 23 visible in any single crosswalk. Thus, when writing our aggregation, we can choose a range constraint of 23, rather than 40, which we would have to use without spatial splitting.

Case 1: Query 3

```
SPLIT urban
    BEGIN 06-01-2019/06:00am END 06-01-2019/06:00pm
    BY TIME 30sec STRIDE 0sec
    BY REGION "crosswalk"
    WITH MASK U2
    INTO urbanChunks;
PROCESS campusChunks USING count_ppl_urban.py TIMEOUT 1sec
    PRODUCING 1 ROWS
    WITH SCHEMA (ppl:NUMBER=0)
    INTO campusTable;
SELECT hour, sum(RANGE(ppl,0,23)) from campusTable
    GROUP BY hour
    CONSUMING eps=1.0;
```

Sources of Inaccuracy. PRIVID introduces two sources of inaccuracy to a query result: (1) intentional noise to satisfy (ρ, K, ϵ)-privacy, and (2) (unintentional) inaccuracies caused by the impact of splitting and masking videos before executing the video processing. Figure 5-1 shows these two sources separately for queries Q1-Q3 (Case 1): the discrepancy between the two curves demonstrates the impact of (2), while the shaded belt shows the relative scale of noise added (1). In summary, the scale of error added by PRIVID allows the final result to preserve the trend of the original.

5.2.2 Case 2: Aggregating Over Multiple Cameras (Q4-Q6)

In this case, we demonstrate (1) PRIVID's ability to express complex queries over multiple cameras and (2) the ability to execute multiple aggregations over the same intermediate tables. We chose 3 representative questions an analyst might seek to answer about the mobility patterns of taxis in a city, using the Porto Taxi Dataset:

- What are the average working hours of taxis?
- How many taxis made the trip from region A to region B in a single day?
- Which region of the city (camera) had the most congestion each day?

Each of these queries requires the same underlying primitive data: a table of taxi appearances per camera. Thus, we can strucutre our PQL query as follows. We first use a `SPLIT` per camera to break the full set of video data into chunks, then use a `PROCESS` to create a table per camera. In contrast to the previous queries, in this case we do care about details of the objects, so a simple count will not suffice. We need to know the license plate of each object so that we can aggregate over the same objects observed in separate cameras. Thus, our executable outputs a row per car observed, with a single column for its license plate. The result is a table of car appearances, one row per appearance (the same taxi may appear in the same camera multiple times).

Then, we can formulate each of our queries as three separate `SELECT` aggregations over the *same* intermediate tables. Each table is tied to the video that created it, so after each aggregation, the budget will be decremented accordingly. Assuming that there is a budget of $\epsilon = 1$ available, we choose to allocate our budget evenly among the queries, using $\frac{1}{3}$ for each.

Case 2: Queries 4-6

```
-- Repeat for portoCam1...portoCam127:
SPLIT portoCam1
    BEGIN 07-01-2013/12:00am END 07-01-2014/12:00am
    BY TIME 15sec STRIDE 0sec
    INTO chunks1;
-- Repeat for chunks1...chunks127:
PROCESS chunks1 USING porto.py TIMEOUT 1sec
    PRODUCING 3 ROWS
    WITH SCHEMA (plate:STRING="")
    INTO table1;

-- Query 4: Average Taxi Working Hours
SELECT avg(avg_shift) FROM
    (SELECT plate,avg(RANGE(shift, [0,16])) FROM
        (SELECT plate,day,(max(chunk)-min(chunk) as shift) FROM
            table10 UNION table27 GROUP BY plate,day(chunk))
    GROUP BY plate LIMIT 300)
CONSUMING eps=0.33;
-- Query 5: # Taxis Traversing Both Locations On Same Day
SELECT day,count(DISTINCT plate) FROM
    (SELECT day,plate FROM
        table10 INNER JOIN table27 ON
        (table10.plate=table27.plate AND table10.day=table27.day)
    )
    GROUP BY day
CONSUMING eps=0.33;
-- Query 6: Camera with highest daily traffic
SELECT argmax(arg=cam, target=avg_daily) FROM
  (SELECT "cam1" as cam, avg(daily) as avg_daily FROM
    (SELECT day,count(DISTINCT plate) as daily FROM
      table1 GROUP BY day))
  UNION
  // ...
  UNION
  (SELECT "cam127" as cam, avg(daily) as avg_daily FROM ...)
CONSUMING eps=0.33;
```

Policy. We chose a ρ for each camera by chosing the maximum amount of time a car was visible at any camera over the course of the entire dataset. We chose a value of $K = 1$, because we expect most cars to pass through each camera only once in a typical day. While taxis likely pass through more often, their existence is not private information. We only seek to hide their movement patterns, which is achieved by using a proper ρ to bound each individual *appearance*. There are too many cameras to include the policies for each inline, so we supply just two here which we will use for the example sensitivity calculations below: $\mathcal{P} = \{(\rho = 45s, K = 1)_{c_1}, (195s, K = 1)_{c_2}\}$

Sensitivity of Q4. We can express this query in relational algebra as follows:

$$\Pi_{\text{Avg(hrs)}}(\sigma_{\text{limit(plates)=300}}({}_{\text{plate,day}}\gamma_{\text{range(chunks)}\in[0,16]}(t_1 \cup t_2)))$$

First, we compute the base sensitivity of each table. The `SPLIT` statement specifies the video will be split into 15 second chunks with 0 stride, and that each chunk will produce a maximum of 3 rows. With this we can compute: $\Delta_{\mathcal{P}}(t_1) = \lceil \frac{(45*\texttt{fps}-1}{15*\texttt{fps}} \rceil + 1 = 4 \cdot 3 = 12$ and $\Delta_{\mathcal{P}}(t_2) = \lceil \frac{195*\texttt{fps}-1}{15*\texttt{fps}} \rceil + 1 = 14 \cdot 3 = 42$. When we combine them with a union, their sensitivities add: $\Delta_{\mathcal{P}}(t_1 \cup t_2) = 12 + 42 = 54$. The `GROUP BY` creates a new table with a row per plate per day, and constrains the range of the aggregate value `shift` to $[0, 16]$ ($\texttt{range}(a, b)$ returns $|b - a|$, i.e., the time between the first and last appearance of a taxi on a given day), but we don't know how many unique plates there might be, so the size $\tilde{C}_s(\gamma(...))$ is unconstrained. We add σ_{limit} to manually enforce a maximum of 300 plates per day (based on the publicly available number of taxis), which gives us a constraint $\tilde{C}_s(\sigma(...)) = 300\text{plates} * 365\text{days} = 109,500$. We now have all the constraints necessary to compute the sensitivity of the average aggregation: $\Delta_{\mathcal{P}}^{\text{AVG}}(R) = \frac{\Delta_{\mathcal{P}}(R)\tilde{C}_r(R,\text{shift},)}{\tilde{C}_s(R)} = \frac{54 \cdot 16}{109{,}500} = 0.0079$. Since Privid uses the Laplace mechanism to add noise, we can use the inverse CDF of the Laplace distribution to bound the expected error based on Δ with a given confidence level. For example, $\mathcal{L}^-1(p = 0.999, u = 0, b = \frac{\Delta}{\epsilon} = \frac{0.0079}{0.33}) \leq 0.15$ hours with 99.9% confidence.

5.2.3 Case 3: Counting Non-Private Objects (Q7-Q9)

For this case, we construct queries similar to Q1-Q3, but increase the query window. Although the scale of noise is the same regardless of query window, the impact is decreased relative to the total. We measure the fraction of trees (non-private objects) that have bloomed in each video. While this query could be done manually for a single camera, it would be cumbersome for a huge network of cameras. Across the entire network, this query could quick identify regions with the best foliage in spring. Relative to Case 1, we achieve high accuracy by using a longer query window of 12 hours (the status of a tree does not change on that time scale), and minimal chunk size (1 frame, no temporal context needed).

Case 3: Q7-Q9

```
SPLIT campus
    BEGIN 06-01-2019/06:00am END 06-01-2019/06:00pm
    BY TIME 1frame STRIDE 1min
    WITH MASK C2
    INTO campusChunks;
PROCESS campusChunks USING trees.py TIMEOUT 1sec
    PRODUCING 1 ROWS
    WITH SCHEMA (frac:NUMBER=0)
    INTO campusTrees;
// Repeat for highway WITH MASK H3
// Repeat for urban WITH MASK U1
SELECT avg(frac) FROM campusTrees
    CONSUMING eps=1;
SELECT avg(frac) FROM highwayTrees
    CONSUMING eps=1;
SELECT avg(frac) FROM urbanTrees
    CONSUMING eps=1;
```

5.2.4 Case 4: Duration of Non-Private Objects (Q10-Q12)

For this case, we demonstrate the utility of the most extreme masks which remove *all* pixels of the frame that ever observe a private object. In particular, we aim to compute the duration that a traffic light stays red in each video. Thus, we can choose a mask which removes all pixels except the traffic lights. This corresponds to a ρ bound of 0, which means that our query can be executed without any noise, and thus can also be executed over a finer temporal granularity (in this case, one result every 10 minutes).

Case 4: Q10-Q12

```
SPLIT campus
    BEGIN 06-01-2019/06:00am END 06-01-2019/06:00pm
    BY TIME 10min STRIDE 0sec
    WITH MASK C3
    INTO campusChunks;
PROCESS campusChunks USING lights.py TIMEOUT 1sec
    PRODUCING 1 ROWS
    WITH SCHEMA (duration_secs:NUMBER=0)
    INTO campusLights;
// Repeat for highway WITH MASK H4
// Repeat for urban WITH MASK U3
SELECT chunk,avg(duration_secs) FROM campusLights;
SELECT chunk,avg(duration_secs) FROM highwayLights;
SELECT chunk,avg(duration_secs) FROM urbanLights;
```

5.2.5 Case 5: Trajectory Queries (Q13)

For this case, we consider a scenario where the analyst needs to observe the entire movement pattern of an individual within a single chunk. In particular, we aim to compute the set of individuals in `campus` that enter from the south and exit at the north. This requires a larger chunk size (relative to Q1-Q3) to ensure that we observe the full trajectory.

Case 5: Q13

```
SPLIT campus
    BEGIN 06-01-2019/06:00am END 06-01-2019/06:00pm
    BY TIME 1min STRIDE 0sec
    INTO campusChunks;
PROCESS campusChunks USING trajectory.py TIMEOUT 1sec
    PRODUCING 1 ROWS
    WITH SCHEMA (ppl:NUMBER=0, trajectory=STRING="")
    INTO campusPpl;
SELECT sum(RANGE(ppl,0,25)) FROM campusPpl WHERE enter=="south" AND exit=="north";
```

Chapter 6

Privid in Practice

In this chapter, we discuss the practical aspects of operationalizing Privid from the perspective of both the Video Owner and the analyst(s). We enumerate the decisions that each party needs to make and provide some suggestions and best practices.

6.1 Video Owner's Perspective

6.1.1 Decisions

The Video Owner must "register" a set of cameras with Privid. For *each* camera, they must supply: (1) a (ρ, K) bound (or more generally a map of masks to bounds), (2) a privacy budget allocation strategy (ϵ), and (3) some metadata describing the scene to analysts (e.g., a short video clip, since they cannot view the camera feed directly). All of this is public to analysts. Below we provide general suggestions for the Video Owner, but ultimately they are responsible for choosing these values. Privid does not generate policies, it only handles enforcing them.

(1) (ρ, K) **bounds.** In most cases, a good starting place is the following: record a representative sample of video from the camera, measure durations of objects of interest (typically cars and people) using off-the-shelf CV algorithms for detection and tracking, and then pick the tightest possible (ρ, K) that would bound all objects. There are a few important considertaions at each step. First, the Video Owner should include as much video as possible in their representative sample. Ideally, it should be at least one week to capture both diurnal patterns and differences between weekdays and weekends. Depending on the scene, there may be other factors to consider about how durations may change over longer periods of time. They could also continue to analyze the distriubtion of durations each day until the bound stabilizes and no new outliers are found. Even once a bound is picked, the Video Owner may want to periodically analyze recent video data to ensure that duration patterns have not changed. The estimation of an object's duration depends on the performance of the detection and tracking algorithms. Each of these algorithms offer a number of hyperparameters that tradeoff different aspects of their performance. We suggest choosing conservative hyperparameters that err on the side of grouping more (and we provide guidance on how to do this in §6.1.2). Finally, there should be a human in the loop to debug this process. The produced value should align with what is reasonable based on the scene. If it is far tighter or looser, the Video Owner should inspect the object tracks and adjust the parameters as necessary. Once they are confident that the object tracks are reasonable, they must also consider whether they want to protect multiple appearances. If the scene allows reliable person re-identification, this process could also be automated. If not, the Video Owner may want to manually pick a value of K based on the scene. For example, if it is a typicaly street, $K = 1$ is probably reasonable. If it overlooks a building, $K = 2$ would protect a single person both entering and exiting. Finally, if it is an office building that the same people visit every day, $K = 2 \cdot 5$ would protect an entire week of their appearances.

To provide better utility for analysts, the Video Owner can offer a menu of static masks that remove some of the scene in exchange for tighter noise bounds than the original policy (which is itself mapped to the empty mask). Algorithm 2 provides an efficient algorithm for generating a list of useful masks and policies. As with the single policy case, the Video Owner should not trust these results blindly, but should use them instead as a starting point and adjust if necessary.

The Video Owner may draw masks manually or generate them automatically, e.g., by analyzing past trends from the camera. In general, we expect masks to be static properties of each scene, dependent only on dynamics of the scene type, rather than behaviors of any individuals. However, it is ultimately the Video Owner's responsibility to ensure any masks it provides do not reveal anything private, such as a person's silhouette. Privid focuses on preventing the leakage of privacy when answering queries. It does not make any guarantees about the mask itself.

(2) Budget ϵ. As in any deployment of DP, the choice of ϵ is subjective. Academic papers commonly use $\epsilon \approx 1$ [51] while recent industry deployments have used $1 < \epsilon < 10$ [57, 22, 32]. In the simplest case, the Video Owner can choose a reasonable global ϵ to initialize the budget for all frames of all cameras. They can choose to lower the budget or even set it to zero for all frames in a particularly sensitive time range. They can also choose to lower or raise it for some cameras with more (e.g., homes) or less sensitive viewing areas. The only Privid-specific consideration for choosing ϵ is that cameras with overlapping fields of view should share the same budget.

(3) Video Clips. The Video Owner should release sample video clip(s)[1] representative of the scene so that analysts can calibrate their executable[2] and query[3] accordingly.

To be explicit, Privid does not provide any privacy guarantees around these clips. Although these video clips do contain the same objects whose privacy we wish to protect using Privid, we believe releasing these clips is a worthwhile trade-off: they result in limited one-time privacy loss (which can be manually vetted and controlled by the Video Owner), and they stand to make the system significantly more useful from the analyst's perspective. This is in line with Privid's overarching goal of thwarting tracking and surveillance; a single appearance of an individual in a fixed clip does not aid a malicious analyst in tracking them outside of this clip.

(3) Other Metadata. Optionally, the Video Owner can release additional information to aid analysts, such as the camera's GPS coordinates, make, or focal length settings.

6.1.2 Estimating Durations Using CV

video	**cos**	**iou**	**age**	**n_init**
campus	0.1, 0.3, **0.5**, 0.7, 0.9	0.1, 0.3, 0.5, **0.7**, 0.9	16, 32, 48, 64, 80, **96**, 112	2, 3, 5, 7, **9**
urban	**0.1**, 0.3, 0.5, 0.7, 0.9	0.1, 0.3, **0.5**, 0.7, 0.9	8, 16, 32, 48, 64, 80, **96**	2, 3, **5**, 7, 9

Table 6-1: Set of hyperparameters used for tuning DeepSORT for the `campus` and `urban` videos. The set of parameters that we ultimately used for our experiments are bolded.

video	**max_age**	**min_hits**	**iou_dist**
highway	240, 480, **720**	3, 5, 7, **9**	**0.1**, 0.3, 0.5, 0.7

Table 6-2: Set of hyperparameters used for tuning SORT for the `highway` video. The set of parameters that we ultimately used for our experiments are bolded.

[1]While a clip is not needed in principle, without it, the analyst "runs blind" and will not have confidence in the correctness of their results.

[2]ML models may perform better when retrained on a particular scene.

[3]For example, queries must specify bounds on the amount of output per chunk, which depend on the amount of activity in the scene.

Estimating duration values for a given scene requires the ability to track individuals in that scene. Unfortunately, even state-of-the-art vision techniques for object tracking are riddled with inaccuracies that stem from occlusion (i.e., line of sight to an object is blocked), illumination, and poor video quality; these challenges are exacerbated in low-quality public surveillance videos. Manual annotation of individuals in video can overcome these challenges but is far from scalable and is difficult to use for real-time video analysis.

We observe that, even though the aforementioned challenges preclude off-the-shelf algorithms from perfectly tracking every individual, their hyperparameters can be trained in a way that generates a reasonably accurate distribution of duration values, which is sufficient for Privid to provide meaningful privacy guarantees.

For each of the three video in our dataset, we first ran object detection using Facebook's Detectron2 [85] library with the included Faster-RCNN model [72]. Using these object detection results, we then manually annotated a subset of video for each camera, producing a ground truth dataset of duration values. Annotation for a video involved recording the exact time each unique individal entered and exited the scene at the second granularity.

Using our ground truth dataset, we then tuned the hyperparameters of a state-of-the-art tracking algorithm called DeepSORT [82] for each camera's video. Our goal was to find the configuration of parameters that produced the distribution of duration values which most closely matched that of the annotated ground truth data. To do this, we ran DeepSORT with all possible combinations of the hyperparameters listed in Table 6-1. For each configuration, we computed the distribution of duration values, and compared it to our ground truth distribution.

In `highway`, we consider cars as the private object rather than people because no people are visible in the video, but a car's license plate, or their combination of make, model and color may be enough to identify an individual. As DeepSORT is specific to tracking people, we used SORT [24] instead. Table 6-2 lists the set of hyperparameters we considered and chose for tuning SORT. In practice, if a video contains both people and cars, the duration distribution analysis should account for both.

6.2 Analyst's Perspective

We first outline the decisions the analyst must make, then explore some of the tradeoffs in more detail in the following subsections.

6.2.1 Decisions

In order to formulate a Privid query the analyst must make the following decisions. For each decision, we provide an example for the query in §4.7.1 (counting cars crossing a virtual line on a highway).

Choose a mask (from the list provided by the Video Owner) based on the query goal. For example, they should select a mask that covers as much of the scene as possible without covering the area near the virtual line. This would significantly reduce the bound by removing parking spots and intersections where objects linger.

Choose a chunk size based on the amount of context needed. A larger chunk size permits more context for each execution of the `PROCESS`, but results in more noise (§4.3). Thus, the analyst should choose the smallest chunk size that captures their events of interest. For example, 1 second is likely sufficient to capture cars driving past a line. If they instead wanted to calculate car speed, they would need a larger chunk size (e.g., 10 seconds) and less restrictive mask to capture more of the car's trajectory.

Choose upper bound on number of output rows per chunk based on the expected (via the video sample) level of activity in each chunk. For counting cars over a short chunk, especially in less busy scenes, each chunk may see 1-2 cars and thus need 1-2 rows. For calculating speed over a larger chunk, especially in more busy scenes, each chunk will see more cars and may need 10 or 100 rows.

Create a `PROCESS` executable. This involves tuning their CV models based on the scene (via the sample video), and combining all tasks into a single executable. For example, their executable may include an object detector to find cars, an object tracker to link them to trajectories, a license plate reader to link cars across cameras or prevent double counting, and an algorithm to compute speed or determine car model.

Choose query granularity and budget. The query granularity and budget are directly proportional to accuracy. Given a fixed value for each, improving one requires worsening another proportionally. We elaborate upon this tradeoff in the following subsection.

6.2.2 Budget-Granularity Tradeoff

Analysts have two main knobs for each query Q to navigate the utility space: (1) the fraction ϵ_Q of the total budget ϵ used by that query, and (2) the duration (granularity)

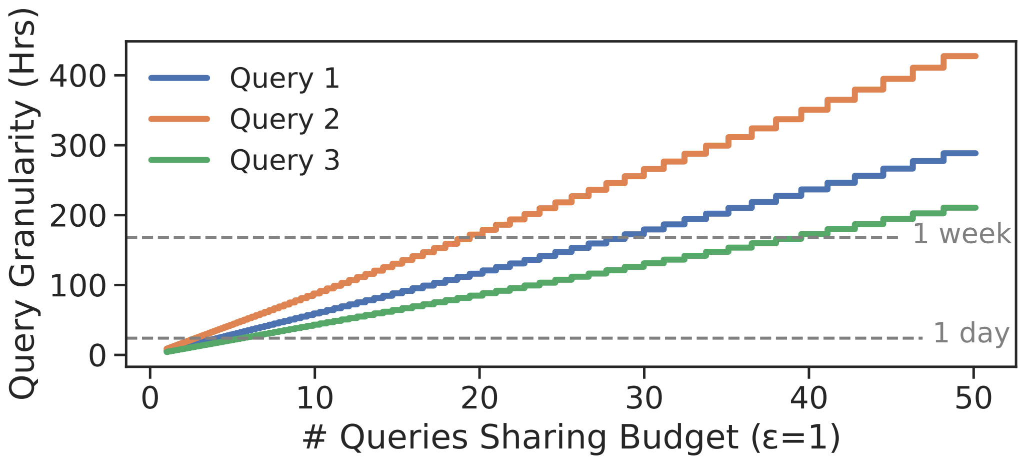


Figure 6-1: Given a fixed query and accuracy target, decreasing the amount of budget used by each query allows more queries to be executed over the same video segment, but requires a proportionally coarser granularity. The x-axis plots the number of queries evenly sharing a budget of $\epsilon = 1$, thus $x = 10$ means 10 instances of the same exact query over the same video segment, each using a budget of $\frac{1}{10}$. We fix the accuracy target to be 99% of values having error $\leq 5\%$.

of each aggregation (i.e., "one value per day for a month" has a granularity of one day). The query budget is inversely proportional to both the query granularity and error (the expected value of noise Privid adds relative to the output range). Thus, to decrease the amount of budget per query (or equivalently, increase the number of queries sharing the budget), an analyst must choose a (temporally) coarser result, a larger expected error bound, or both. Figure 6-1 shows that, for example, 5 instances of Query 3 could release results daily or 40 instances of Query 3 could release results weekly, while achieving the same expected accuracy. Figure 6-2 shows that, for example, 20 separate instances of Query 1 ($x = 20$) executed over the same target video could each expect 4.8% error if they release one result daily, 0.7% error if they release one weekly, or 0.16% error if they release one monthly. Importantly, this tradeoff is transparent to analysts: Figures 6-1 and 6-2 rely only on information that is publicly available to analysts and did not require executing any queries.

6.2.3 Best Practices for Query Writing

In the following subsections we provide strategies for designing queries that optimize utility.

Counting Unique Objects. The main limitation of Privid's query interface is the inability to write queries that maintain state across separate chunks. However, in

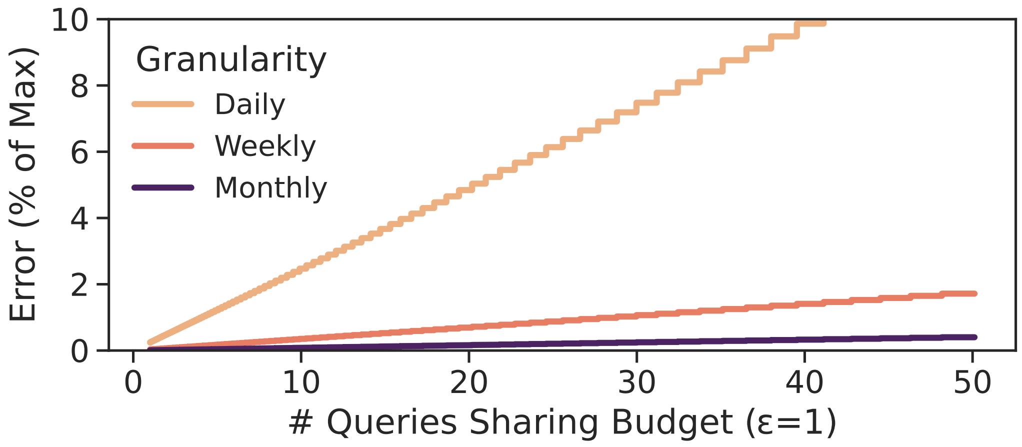


Figure 6-2: Given a fixed query and granularity, decreasing the amount of budget used by each query allows more queries to be executed over the same video segment, but results in proportionally higher error. The x-axis is the same as Figure 6-1. Each line corresponds to Q1 using a different granularity. The y-axis plots the error for 99% of values. Error is the amount of noise added relative to the maximum query output. For example, in Q4, the final output is the average number of working hours in the range [0,16]. Thus an error of 1% would mean the noisy result is within 0.16 hours of the true result.

most cases this does not preclude queries, it simply requires them to be expressed in a particular way. One broad class of such queries are those that operate over *unique* objects.

Consider a query that wants to count the total number of cars or people that pass a camera. A straightforward implementation might detect `car` or `people` objects, output one row for each object, and then compute an aggregate count of the number of rows. However, if a car enters the camera view in chunk i and is last visible in chunk $i + n$, the `PROCESS` table will include n rows for the same car instead of the expected 1.

The are two possible cases: either the objects of interest have globally unique identifiers or not. If these identifiers exist and are reliable, such as the license plate on a car, we can adjust our query to leverage this identifier: our executable can incorporate a license plate reader which outputs a row with a `plate` string for each car, and then we can `count(DISTINCT plate)` in the `SELECT` (as in §4.7.1). This `DISTINCT` directive will remove any duplicates in the license plates before executing the count. This will make our count more accurate, but it *does not* actually change the number of plates a (ρ, K)-bounded event could impact, and thus does not change the sensitivity or require adding any more noise.

For people we can use similar stretgies. If faces are recognizable and can be represented in a vector format, we can output this vector as a row. We can then use a user-defined function to define whether or not two vectors are close enough to represent the same face.

When these identifiers are not easily idenfiable or are not reliable (e.g., the angle of the camera does not capture the plate or face), a general strategy is to adjust the query and redefine our event of interest such that it is only visible in a single chunk.

For example, if the goal is simply to count the number of unique pedestrians that a camera observes, rather than counting all individuals that are visible at any time, we can decide to (a) only count people that *enter* the scene, or (b) only count people that cross a virtual line. Consider case (b). We count any object track that crosses the virtual line during the chunk, except during the first frame of the chunk. If the track crosses it during the first frame, then it means it must have been observed entering the link during the previous chunk. The result is that all individuals that "appear" will contribute a total count of 1. If the query chunks are strided such that chunks overlap and any given frame can be in at most s frames, then we can adapt the same strategy, but instead output $\frac{1}{s}$ for each individual that enters.

Finally, if the event of interest requires a lot of context, e.g. counting the set of individuals that enter at one side of the frame and leave at the other, we can set the chunk size to the typical amount of time it takes to cross the frame and then use a stride to make sure each individual's trajectory is captured in at least one full chunk. We only count it if it the full movement happens in a chunk. For all the partially overlapping chunks where the trajectory is only partially visible, we do not count it at all.

Spread queries over time. Suppose you have access to one month of video from a highway camera and want to answer to entirely separate queries: (1) the distribution of car types, and (2) the average speed per car type. This month has ϵ budget remaining. Running both of these queries over the same portion of video would require splitting this budget among the queries, using at most $\frac{\epsilon}{2}$ for each. Instead, if we expect these distributions to remain the same over time, we can execute them over disjoint time frames (e.g. the first query over the first two weeks, and the second query over the second two weeks) so that we can use the full ϵ budget for each. The downside is that each query is aggregating over less data, and thus may be less representative of the true distribution. To combat this, we can break each of our queries into two parts, one to test the consistency of the distribution over time, and another to actually release the distribution itself. First, we compute the distribution over both the two-week period and the month. Then, we compute the similarity between these distributions and only *release* a binary answer of whether or not the similarity is above a certain threshold (not the distributions themselves). Due to the low sensitivty of this threshold query, we can answer it with high accuracy using only a small portion of the budget (e.g. $\frac{\epsilon}{10}$). We can repeat this for the second query as well, and we are left with $\frac{8\epsilon}{10}$ budget remaining across the entire both. If the distribution is consistent over time, now we can release the full distribution for the first query over the first two week period, and the full distribution for the second query over the second two week period. In each case we can use $\frac{8\epsilon}{10}$ rather than $\frac{\epsilon}{2}$, which will give lower noise bounds.

6.3 Discussion: Social Implications

An important question to ask ourselves is: how would our society be impacted if Privid were *widely* deployed? In other words, if there was an expectation that all (or most) cameras we pass on a daily basis were being actively analyzed using Privid, would that cause us to act differently? While we do not have a definitive or exhaustive answer, we raise one potential concern that is worth further consideration: given that privacy is tied to duration of visibility, it could cause people to become anxious about staying in one place for too long, or doing any activity for too long, for fear that it could be detected.

Chapter 7

Conclusion and Future Work

Today, copious amounts of video data are recorded and stored, not only by surveillance cameras, but also by all of our dashcams and smartphones. The field of computer vision has given us an amazing opportunity to make sense of all this data, but much of its promise is stunted by serious privacy concerns, and rightfully so. In order to move forward, it is critical that we have systems that incorporate privacy as a first principle, not in isolation, but alongside utility and generality. In this dissertation, we presented Privid, a general-purpose video analytics system that aims to provide this balance. In particular, Privid exposes a query interface that is familiar to analysts, which allows them to use their existing vision models without retraining. At the same time, our privacy definition has a clear interpretation, and Privid provides a rigorous guarantee that it satisfies this definition, which can serve as the foundation for reliable privacy policies that people can trust.

My hope is that Privid will provide a way forward for video analytics to progress without eroding our privacy in the process. That being said, Privid in its current state represents only a first step, and there are still opportunities to improve its usefulness for analysts. In particular, in this work we focused on the temporal aspect of videos, but there are other dimensions along which videos could be individually processed, such as by spatial region—both at the micro level by dividing individual frames into smaller pieces, and at the macro level by combining frames from multiple cameras at the same time instant—or by color spectrum. Any other processing dimension could be incorporated into Privid's pipeline and would expand the set of queries it supports.

More broadly, I hope that this work adds a small piece of evidence that it is indeed possible to achieve a healthy balance between privacy and utility in practice. I hope this encourages more systems researchers and practitioners to consider incorporating privacy as a first principle in their work.

Bibliography

[1] Absolutely everywhere in beijing is now covered by police video surveillance. https://qz.com/518874/.

[2] Are we ready for ai-powered security cameras? https://thenewstack.io/are-we-ready-for-ai-powered-security-cameras/.

[3] British transport police: Cctv. http://www.btp.police.uk/advice_and_information/safety_on_and_near_the_railway/cctv.aspx.

[4] Can 30,000 cameras help solve chicago's crime problem? https://www.nytimes.com/2018/05/26/us/chicago-police-surveillance.html.

[5] Data generated by new surveillance cameras to increase exponentially in the coming years. http://www.securityinfowatch.com/news/12160483/.

[6] Detection leaderboard. https://cocodataset.org/#detection-leaderboard.

[7] Epic domestic surveillance project. https://epic.org/privacy/surveillance/.

[8] Paris hospitals to get 1,500 cctv cameras to combat violence against staff. https://bit.ly/2OYiBz2.

[9] Powering the edge with ai in an iot world. https://www.forbes.com/sites/forbestechcouncil/2020/04/06/powering-the-edge-with-ai-in-an-iot-world/.

[10] Video analytics applications in retail - beyond security. https://www.securityinformed.com/insights/co-2603-ga-co-2214-ga-co-1880-ga.16620.html/.

[11] The vision zero initiative. http://www.visionzeroinitiative.com/.

[12] What's wrong with public video surveillance? https://www.aclu.org/other/whats-wrong-public-video-surveillance, 2002.

[13] Abuses of surveillance cameras. http://www.notbored.org/camera-abuses.html, 2010.

[14] Mission creep-y: Google is quietly becoming one of the nation's most powerful political forces while expanding its information-collection empire. https://www.citizen.org/wp-content/uploads/google-political-spending-mission-creepy.pdf, 2014.

[15] Mission creep. https://www.aclu.org/other/whats-wrong-public-video-surveillance, 2017.

[16] How retail stores can streamline operations with video content analytics. https://www.briefcam.com/resources/blog/how-retail-stores-can-streamline-operations-with-video-content-analytics/, 2020.

[17] The mission creep of smart streetlights. https://www.voiceofsandiego.org/topics/public-safety/the-mission-creep-of-smart-streetlights/, 2020.

[18] Video analytics traffic study creates baseline for change. https://www.govtech.com/analytics/Video-Analytics-Traffic-Study-Creates-Baseline-for-Change.html, 2020.

[19] Pest. the elegant parser. https://github.com/pest-parser/pest, 2022.

[20] Paarijaat Aditya, Rijurekha Sen, Peter Druschel, Seong Joon Oh, Rodrigo Benenson, Mario Fritz, Bernt Schiele, Bobby Bhattacharjee, and Tong Tong Wu. I-pic: A platform for privacy-compliant image capture. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, page 235–248, New York, NY, USA, 2016. Association for Computing Machinery.

[21] Ganesh Ananthanarayanan, Yuanchao Shu, Mustafa Kasap, Avi Kewalramani, Milan Gada, and Victor Bahl. Live video analytics with microsoft rocket for reducing edge compute costs, July 2020.

[22] Apple Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 1(8), 2017.

[23] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, and Sam Madden. Miris: Fast object track queries in video. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 1907–1921, New York, NY, USA, 2020. Association for Computing Machinery.

[24] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[25] Sam Biddle. https://theintercept.com/2023/05/15/abortion-surveillance-dataminr/, May 2023.

[26] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 3361–3369, Washington, DC, USA, 2015. IEEE Computer Society.

[27] Frank Cangialosi, Neil Agarwal, Venkat Arun, Srinivas Narayana, Anand Sarwate, and Ravi Netravali. Privid: Practical,{Privacy-Preserving} video analytics queries. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 209–228, 2022.

[28] Ankur Chattopadhyay and Terrance E Boult. Privacycam: a privacy preserving camera using uclinux on the blackfin dsp. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[29] S-CS Cheung, M Vijay Venkatesh, Jithendra K Paruchuri, Jian Zhao, and Thinh Nguyen. Protecting and managing privacy information in video surveillance systems. *Protecting Privacy in Video Surveillance*, pages 11–33, 2009.

[30] Ji Dai, Behrouz Saghafi, Jonathan Wu, Janusz Konrad, and Prakash Ishwar. Towards privacy-preserving recognition of human activities. In *2015 IEEE international conference on image processing (ICIP)*, pages 4238–4242. IEEE, 2015.

[31] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. 2004.

[32] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3571–3580. Curran Associates, Inc., 2017.

[33] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284, Berlin, Heidelberg, March 2006. Springer.

[34] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

[35] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[36] Geoffrey A. Fowler. Google promised to delete sensitive data. it logged my abortion clinic visit. `https://www.washingtonpost.com/technology/2023/05/09/google-privacy-abortion-data/`, May 2023.

[37] Isha Ghodgaonkar, Subhankar Chakraborty, Vishnu Banna, Shane Allcroft, Mohammed Metwaly, Fischer Bordwell, Kohsuke Kimura, Xinxin Zhao, Abhinav Goel, Caleb Tung, et al. Analyzing worldwide social distancing through large-scale computer vision. *arXiv preprint arXiv:2008.12363*, 2020.

[38] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[39] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 269–286, 2018.

[40] Samvit Jain, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, and Joseph E. Gonzalez. Scaling Video Analytics Systems to Large Camera Deployments. In *ACM HotMobile*, 2019.

[41] Samvit Jain, Xun Zhang, Yuhao Zhou, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Victor Bahl, and Joseph Gonzalez. Spatula: Efficient cross-camera video analytics on large camera networks. In *ACM/IEEE Symposium on Edge Computing (SEC 2020)*, November 2020.

[42] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 253–266. ACM, 2018.

[43] Noah Johnson, Joseph P Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539, 2018.

[44] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. *IEEE Transactions on Information Theory*, 63(6):4037–4049, 2017.

[45] Daniel Kang, Peter Bailis, and Matei Zaharia. Blazeit: optimizing declarative aggregation and limit queries for neural network-based video analytics. *Proceedings of the VLDB Endowment*, 13(4):533–546, 2019.

[46] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment*, 10(11):1586–1597, 2017.

[47] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE transactions on computational imaging*, 2(2):109–122, 2016.

[48] Georgios Kellaris, Stavros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12):1155–1166, 2014.

[49] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Privatesql: A differentially private sql query engine. *Proc. VLDB Endow.*, 12(11):1371–1384, July 2019.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[51] Y.-H. Kuo, C.-C. Chiu, D. Kifer, M. Hay, and A. Machanavajjhala. Differentially private hierarchical count-of-counts histograms. *Proceedings of the VLDB Endowment*, 11.11:1509—1521, 2018.

[52] Karen Lander, Vicki Bruce, and Harry Hill. Evaluating the effectiveness of pixelation and blurring on masking the identity of familiar faces. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, 15(1):101–116, 2001.

[53] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5325–5334, June 2015.

[54] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. SIGCOMM ’20, page 359–376, New York, NY, USA, 2020. Association for Computing Machinery.

[55] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, July 2017.

[56] Peng Liu, Bozhao Qi, and Suman Banerjee. Edgeeye: An edge service framework for real-time intelligent video analytics. In *Proceedings of the 1st international workshop on edge systems, analytics and networking*, pages 1–6, 2018.

[57] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, 2008.

[58] Frank D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’09, page 19–30, New York, NY, USA, 2009. Association for Computing Machinery.

[59] Luis Moreira-Matias, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402, 2013.

[60] Mozilla. Shady mental health apps inch toward privacy and security improvements, but manystill siphon personal data. `https://foundation.mozilla.org/en/blog/shady-mental-health-apps-inch-toward-privacy-and-security-improvements-but-many-still-siphon-personal-data/`, May 2023.

[61] Athira Nambiar, Alexandre Bernardino, and Jacinto C Nascimento. Gait-based person re-identification: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–34, 2019.

[62] Akshay Narayan, Frank Cangialosi, Deepti Raghavan, Prateesh Goyal, Srinivas Narayana, Radhika Mittal, Mohammad Alizadeh, and Hari Balakrishnan. Restructuring endpoint congestion control. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 30–43, 2018.

[63] Carman Neustaedter, Saul Greenberg, and Michael Boyle. Blur filtration fails to preserve privacy for home-based video conferencing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 13(1):1–36, 2006.

[64] Elaine M Newton, Latanya Sweeney, and Bradley Malin. Preserving privacy by de-identifying face images. *IEEE transactions on Knowledge and Data Engineering*, 17(2):232–243, 2005.

[65] Mark S Nixon, Tieniu Tan, and Rama Chellappa. *Human identification based on gait*, volume 4. Springer Science & Business Media, 2010.

[66] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: the case of aes. In *Cryptographers' track at the RSA conference*, pages 1–20. Springer, 2006.

[67] José Ramón Padilla-López, Alexandros Andre Chaaraoui, and Francisco Flórez-Revuelta. Visual privacy protection methods: A survey. *Expert Systems with Applications*, 42(9):4177–4195, 2015.

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[69] Colin Percival. Cache missing for fun and profit, 2005.

[70] Rishabh Poddar, Ganesh Ananthanarayanan, Srinath Setty, Stavros Volos, and Raluca Ada Popa. Visor: Privacy-preserving video analytics as a cloud service. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1039–1056, 2020.

[71] Nisarg Raval, Ashwin Machanavajjhala, and Landon P Cox. Protecting visual secrets using adversarial nets. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1329–1332. IEEE, 2017.

[72] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[73] Indrajit Roy, Srinath TV Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and privacy for mapreduce. In *NSDI*, volume 10, pages 297–312, 2010.

[74] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. Fawkes: Protecting privacy against unauthorized deep learning models. In *Proceedings of the 29th USENIX Security Symposium*, 2020.

[75] Hao Sheng, Keniel Yao, and Sharad Goel. Surveilling surveillance: Estimating the prevalence of surveillance cameras with street view data. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 221–230, 2021.

[76] J. Stanley and American Civil Liberties Union. *The Dawn of Robot Surveillance: AI, Video Analytics, and Privacy.* American Civil Liberties Union, 2019.

[77] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 3476–3483, Washington, DC, USA, 2013. IEEE Computer Society.

[78] Verizon. Smart communities / vision zero pilot statement of work. `https://www.boston.gov/sites/default/files/file/document_files/2016/11/bostonverizonpilotsowexecutable11.17.16.docx_.pdf`, 2016.

[79] Han Wang, Yuan Hong, Yu Kong, and Jaideep Vaidya. Publishing video data with indistinguishable objects. *Advances in database technology : proceedings. International Conference on Extending Database Technology*, 2020:323 – 334, 2020.

[80] Han Wang, Shangyu Xie, and Yuan Hong. Videodp: A universal platform for video analytics with differential privacy. *arXiv preprint arXiv:1909.08729*, 2019.

[81] Junjue Wang, Brandon Amos, Anupam Das, Padmanabhan Pillai, Norman Sadeh, and Mahadev Satyanarayanan. A scalable and privacy-aware iot service for live video analytics. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 38–49. ACM, 2017.

[82] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[83] Kok-Seng Wong, Nguyen Anh Tu, Anuar Maratkhan, and M.Fatih Demirci. A privacy-preserving framework for surveillance systems. In *2020 the 10th International Conference on Communication and Network Security*, ICCNS 2020, page 91–98, New York, NY, USA, 2021. Association for Computing Machinery.

[84] Hao Wu, Xuejin Tian, Minghao Li, Yunxin Liu, Ganesh Ananthanarayanan, Fengyuan Xu, and Sheng Zhong. Pecam: Privacy-enhanced video streaming and analytics via securely-reversible transformation. In *ACM MobiCom*, October 2021.

[85] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[86] Zhenyu Wu, Zhangyang Wang, Zhaowen Wang, and Hailin Jin. Towards privacy-preserving visual recognition via adversarial training: A pilot study. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 606–624, 2018.

[87] Xiaoyi Yu, Kenta Chinomi, Takashi Koshimizu, Naoko Nitta, Yoshimichi Ito, and Noboru Babaguchi. Privacy protecting visual processing for secure video surveillance. In *2008 15th IEEE International Conference on Image Processing*, pages 1672–1675. IEEE, 2008.

[88] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. Live video analytics at scale with approximation and delay-tolerance. In *NSDI*, volume 9, page 1, 2017.

[89] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417, 2017.