Title
Balancing Behavioral Privacy and Information Utility in Sensory Data Flows

Author
Chakraborty, Supriyo

Publication Date
2014

Peer reviewed|Thesis/dissertation

University of California
Los Angeles

Balancing Behavioral Privacy and Information Utility in Sensory Data Flows

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical Engineering

by

Supriyo Chakraborty

2014

Abstract of the Dissertation

Balancing Behavioral Privacy and Information Utility in Sensory Data Flows

by

Supriyo Chakraborty

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2014

Professor Mani Srivastava, Chair

The democratization of computing and sensing through smart phones and embedded devices has led to widespread instrumentation of our personal and social spaces. The sensor data thus collected, has embedded in them minute details of our daily life. On the one hand, this has enabled a multitude of exciting applications where decisions at various time-scales are driven by inferences that are computationally derived from the shared sensory information and used for purposes such as targeted advertisements, behavior tailored interventions and automated control. On the other hand, the ability to derive rich inferences about user behaviors and contexts and their use in critical decision making also present various concerns of personal privacy. Prior approaches to handling the privacy concerns have often been ad hoc and focused on disassociating the user identity from the shared data, thus preventing an adversary from tracing a sensitive inference back to the user. However, in many application domains (e.g., mHealth, insurance) user identity is an inalienable part of the shared data. In such settings, instead of identity privacy, the focus is on the more general inference privacy problem, pertaining to the privacy of sensitive inferences that can be derived from the shared sensor data. The objective of this research has been to develop a principled understanding of the inference privacy problem and design formalisms, algorithms,

and system mechanisms to effectively address it.

The contributions of this dissertation are multi-fold. First, using information-theoretic notions we formulate the inference privacy problem in terms of a whitelist of utility providing allowed inferences, and a blacklist of sensitive inferences. We define utility and privacy parameters, derive bounds on the feasible region spanned by these parameters, and provide constructive schemes for achieving the boundary points of the feasible region. Second, using insights from the theoretical exploration, we design and implement ipShield, a privacy-enforcing system by modifying the Android OS. ipShield, is a step towards reducing the user burden of configuring fine-grained privacy policies. It does so by changing the basic privacy abstraction, from access control on sensors to privacy preferences over higher level possible inferences. The user preferences are then used by a rule recommender to auto-generate privacy rules on sensors. Finally, we present iDeceit, a framework that implements model-based plausible falsification of sensor data to protect the privacy of sensitive inferences while maximizing the utility of the shared data. A graphical model is used to capture the temporal and spatial patterns that exists in user behavior. The model is then used, together with privacy and utility metrics and a novel plausibility metric, to generate falsified data stream that conforms to typical user-behavior ensuring perfect privacy. Extensive evaluation results are detailed for both ipShield and iDeceit to validate their efficiency and feasibility on mobile platforms.

The dissertation of Supriyo Chakraborty is approved.

Gregory Pottie

Suhas Diggavi

Mario Gerla

Mani Srivastava, Committee Chair

University of California, Los Angeles

2014

Dedicated to the loving memory of my mother who instilled in me the value of education, and gave me the strength and courage to always pursue my dreams.

Table of Contents

List of Figures

List of Tables

Acknowledgments

When I joined NESL, about six years ago, I had an isolated data-network centric view of the world. So, when my advisor, Mani Srivastava, introduced me to the world of information privacy, the transition seemed daunting at first. The transformation process that included the broadening of my knowledge horizon, inculcating the ability to think critically and consistently, and finally the carving out of a space where I could make meaningful contributions, was slow and often frustrating. I am deeply indebted to Mani for his unflinching support and guidance throughout this entire journey. He provided me with all the freedom and flexibility in my research that I could want and was patient with me as I explored a diverse range of research interests. His insights and insistence on simplicity (emphasized during discussions as *things should be simple but not simpler*) has always ensured that I did not stray. In the past few years, he has given me a tremendous amount, and no matter how hard I try, I can only thank him for a very small subset.

I am thankful to Greg Pottie, Suhas Diggavi and Mario Gerla for serving on my thesis committee. During my interactions with Greg I was often surprised at his ability to quickly point out critical areas of my research that required more thought. His questions always motivated me to think of alternate solution approaches. With his friendly skepticism and insistence on detail, Suhas's sharp questions forced me to think harder about the relevance of the problem I was trying to solve and the practical applicability of the solution. I always enjoyed my meetings with Mario Gerla as embedded in those conversations were ideas instrumental in shaping and diversifying my thought process. I would also like to thank Lara Dolecek for making me realize the importance of planning and the need to pay attention to details in research.

I would like to thank all my collaborators from the International Technology Alliance (ITA) especially Lance Kaplan, Chatschik Bisdikian, Mudhakar Srivatsa

and Raghu Ganti. The time I spent with them during my internships at IBM Research and later provided me with a much broader perspective on the privacy problem. I became aware of the rich body of work that relied on uncertainty management for both control of information flow and alternately for effective decision making.

In the past six years, while working at CENS and NESL, I have had the privilege of interacting with some exceptionally smart people. Academically, I have co-authored papers with them, and have profited immensely from their suggestions, and high standards. However, the most important aspect of these interactions has been the lasting friendships that we have forged. Graduate life would never have been this fulfilling and fun without them being around. I would like to thank Zainul Charbiwala, Thomas Schmid, Roy Shea, Rahul Balani, Nabil Hajj Chehade, Sadaf Zahedi, Kasturi Raghavan, Nicolas Bitouze, Arijit Khan, George Tychogiorgos, Paul Martin, Henry Herman, Newton Troung, Chenguang Shen, Haksoo Choi, Yasser Shoukri, and Lucas Wanner for all their time, support and encouragement throughout these years.

My work in graduate school would not have been possible without the unconditional support from my wife Rituparna. She has always stood beside me as a rock and have taken care of every aspect of our life. Finally, yet most importantly, I would like to dedicate this work to my father, who while enduring all hardships in his life have ensured that my dreams are never unfulfilled. Without his support I would have never come this far.

Vita

1997–2001	B.Tech (Computer Science), NIT Warangal, India. North Eastern Council Merit Scholarship Award.
2001–2003	Member Technical Staff, HCL Technologies, India.
2003–2006	M.Tech (Electrical Engineering), IIT Bombay, India. Research Assistant, EE Department, IIT Bombay.
2006–2008	Software Engineer, Cisco Systems, Bangalore, India.
2008–2009	UCLA Henry Samueli Fellowship Award.
2009–2010	Teaching Assistant. EE Department, UCLA
2010–2011	Summer Internship at IBM T.J. Watson Research Lab. Teaching Assistant. EE Department, UCLA
2011–2012	Qualcomm Innovation Fellowship Award. Summer Internship at IBM T.J. Watson Research Lab.
2012–2013	Dissertation Year Fellowship Award.
2008–2014	Ph.D. (Electrical Engineering), UCLA.

Publications

Supriyo Chakraborty, Chenguang Shen, Kasturi Raghavan, Matt Millar, Mani Srivastava, ipShield: A Framework For Enforcing Context-Aware Privacy, *In*

11th USENIX Symposium on Networked Systems Design and Implementation, USENIX NSDI, 2014.

Supriyo Chakraborty, Nicolas Bitouze, Mani Srivastava, Lara Dolecek, Protecting Data Against Unwanted Inferences, *In IEEE Information Theory Workshop*, 2013.

Supriyo Chakraborty, Kasturi Raghavan, Matthew Johnson, Mani Srivastava, A Framework for Context-Aware Privacy of Sensor Data on Mobile Systems, *In 14th Workshop on Mobile Computing Systems and Applications, HotMobile*, 2013.

Supriyo Chakraborty, Kasturi Rangan Raghavan, Mani Srivastava, Chatschik Bisdikian, Lance Kaplan, Balancing Value and Risk In Information Sharing Through Obfuscation, *In 15th International Conference on Information Fusion, IEEE FUSION*, 2012.

Supriyo Chakraborty, Kasturi Rangan Raghavan, Mani Srivastava, Chatschik Bisdikian, Lance Kaplan, An Obfuscation Framework for Controlling Value of Information During Sharing, *In IEEE Statistical Signal Processing Workshop*, 2012.

Supriyo Chakraborty, Haksoo Choi, Mani B. Srivastava, Demystifying Privacy in Sensory Data: A QoI Based Approach, *In IEEE Information Quality and Quality of Service (IQ2S) (with IEEE PERCOM)*, 2011.

Supriyo Chakraborty, Zainul M Charbiwala, Haksoo Choi, Kasturi Rangan Raghavan, Mani B Srivastava, Balancing Behavioral Privacy and Information Utility in Sensory Data Flows, *In Pervasive and Mobile Computing Journal*, 2012.

CHAPTER 1

The Predicament of Privacy

The canonical privacy problem for databases can be summed up as: given a collection of personal records from individuals, how do we disclose either the data or "useful" function values such as correlations, population characteristics computed over the data, without revealing any individual information. This notion of absolute privacy is analogous to the principle of semantic security (which itself is the computational complexity counterpart of Shannon's perfect secrecy concept) formulated for cryptosystems [GM82]. However, the notion of *utility* associated with the shared data in conjunction with adversarial access to auxiliary information makes it impossible to design a scheme which will achieve absolute privacy [Dwo06]. Current research in database privacy has thus evolved into a study of the tradeoff involving degradation in the quality of information shared, owing to privacy concerns, and the corresponding effect on the quality of service [LL09, SRP13].

While database privacy has been extensively studied over the past decade, the proliferation of smartphones with embedded and wireless connected wearable sensors has added a new dimension to the problem. Smartphones today are capable of tracking our locations and social neighborhoods, monitoring physiological markers, and learning about our evolving social dynamics. The personal sensor data, richly annotated with both temporal and spatial information, is in turn acquired by a growing ecosystem of often untrusted context-aware third-party apps [sag], to provide us with *personalized* app experiences

such as behavior-tailored insurance plans, mobile health (mHealth) diagnostics and customized recommendations to enrich our social and personal interactions (or targeted advertising). We refer to the benefit to the user of such personalization as *utility*. However, the same data can also be used to make sensitive inferences such as addictions, stress, emotions, passwords, travel trajectories [RGK11, RMM10, CC11, LFR12, PRH11, RAP11, MVB12, Kru09, HON12] – ones the user would want to keep *private*. Sensory data also presents unique privacy challenges. Due to high temporal and spatial granularity, strong correlation between samples, and different modalities, sensory data offers great potential for data mining while making it harder to obfuscate and preserve privacy.

Thus, from the users' perspective there is a tension that exists between the amount of data she would want to share to obtain the maximum utility from the apps and services while at the same time maintaining the desired level of privacy and it is this tension that forms the crux of the privacy problem.

1.1 Privacy: Reality or Myth

Is privacy something that people really care about? Surveys conducted to understand user privacy expectations [RGK11, FHE12, Kru09] summarize interesting opinions about privacy from multiple independent studies. While people in general are oblivious to privacy violations and amenable to sharing their data, the perception quickly morphs into one of concern when apprised of the various sensitive inferences that can be drawn and the resulting consequences.

Recent fiascos have further established privacy as an important sharing constraint. Examples include the de-anonymization of the publicly released AOL search logs [aol] and the movie-rating records of Netflix subscribers [NS08]. The large datasets in question were released to enable data-mining and collaborative filtering research. However, when combined with auxiliary information the

anonymized datasets were shown to reveal identity information of individual users.

Different types of sensory data have also been exploited for privacy violations. For example, households in the U.S. are being equipped with smart meters to collect temporally fine-grained report of energy consumption. This will allow utility to better estimate the domestic power consumption leading to optimized distribution and control of the grid. However, as shown in [MSF10, RMX12], several unintended and sensitive inferences such as occupancy and lifestyle patterns of the occupants can be made from the data in addition to total power consumption. Smart meter data has been used to infer multimedia context in [GJL12]. In fact, privacy has been identified as a major challenge in fine-grained monitoring of residential spaces [KSS09].

Similarly, in medical research the continuous physiological data collected by wearable sensors can be used to infer potentially sensitive information such as smoking or drinking habits, food preferences. While "informed consent" of the data source is the currently used sharing policy, it can be easily overlooked causing privacy violations as exemplified in [dna]. The DNA information, collected from blood samples of a particular tribe, was originally meant for type-2 diabetes research. But was later used to further research in schizophrenia - a condition stigmatized by the tribe - causing extreme anguish and sense of dissatisfaction. Recently, de-anonymization of personal genomes was done using surnames [GMG13].

Activity recognition algorithms [BI04, RMB10] are used by various fitness and wellness apps to infer the users' Transportation Mode (e.g., predict one of three labels: *walking*, *motorized* or *still*). For example, the Ambulation app in [RMB10] combines accelerometer and GPS data to infer the labels with over 90% accuracy. However, data collected for inferring Transportation Mode can also be used to infer other labels sensitive to the user. For example, the same accelerometer when combined with gyroscope data can be used to infer Onscreen Taps and capture keystrokes on the soft keyboard [MVB12] (and also Location [HON12]) with over

80% accuracy. This could lead to leak of sensitive information like password and PIN entered on the phone.

Thus, privacy threat during data disclosure is real and unless adequate mitigation steps are taken it could cause a delay in the adoption of various ubiquitous sensing based applications.

1.2 Privacy Problem Characterization

Depending on how data is shared by the source we can group the various privacy problems into two broad classes [CCS11]:

Identity Privacy: The data is syntactically sanitized by stripping it of personally identifiable information (PII) before sharing - a process called anonymization. The shared data is intended for research pertaining to population-level statistics. However, privacy violation occurs when the data in presence of auxiliary information is de-anonymized to reveal identity information. Netflix [NS08] and AOL [aol] fiascos fall under this category.

There are two important challenges in using PII based anonymization. First, the definition of PII is inadequate for many types of datasets [NS10], including sensory data. For example, while sharing sanitized location traces, the location data itself can be used to re-identify an individual. Hence, it is hard to clearly demarcate the PII's from the other attributes shared. Second, it is assumed that the non-PII attributes cannot be linked to an individual record. However, auxiliary information has been used along with non-PII attributes to de-anonymize large datasets [NS08, Ohm09].

Inference Privacy: For this class of problems the source's identity is not concealed in the shared data. Instead, there exists a specific set of inferences which the source wants to protect. For example, in Section 1.1, data is shared for inferring the Transportation Mode. These desirable inferences form a *whitelist*

which the user wants to allow. However, the same data can be used to infer keystrokes and sensitive locations – inferences sensitive to the user. The sensitive inferences form the blacklist which the user wants to keep private. The privacy problem is to design a system which will take as input the whitelist and blacklist of inferences and translate them into privacy actions on the shared sensors such that the conditions on the lists are satisfied.

1.3 Our Contribution

In this dissertation, we focus on the inference privacy problem and make the following primary contributions.

1.3.1 Protecting Data Against Unwanted Inferences

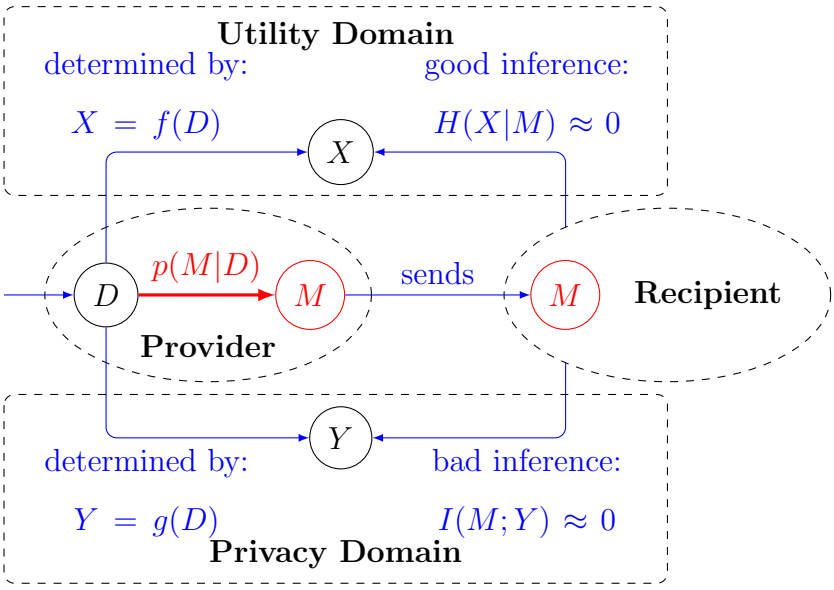


Figure 1.1: The inference privacy problem where the utility domain consists of the whitelist and the privacy domain contains the blacklist.

In Chapter 3, we study the competing goals of utility and privacy as they arise when a provider delegates the processing of its personal information to a recipient

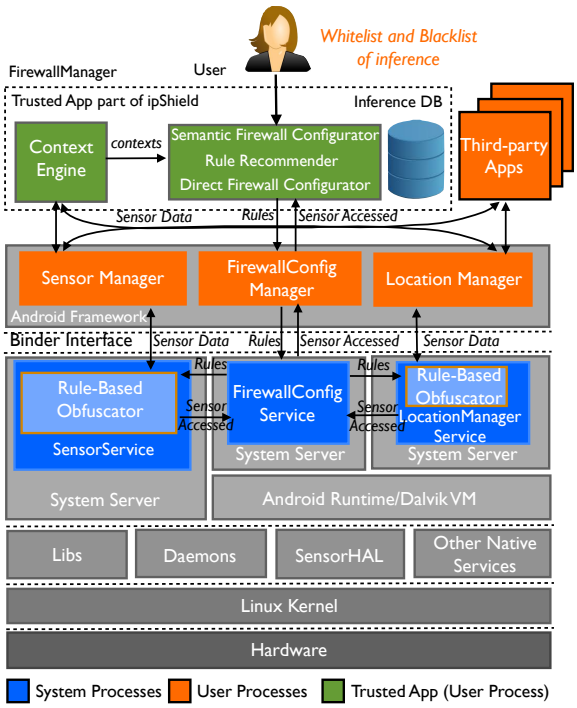


Figure 1.2: ipShield: A privacy enforcing system implemented on the Android OS.

who is better able to handle this data. A whitelist describes the inferences that are desirable, i.e., providing utility and a blacklist describes the unwanted inferences which the provider wants to keep private (see Fig. 1.1). We formally define utility and privacy parameters using elementary information-theoretic notions and derive a bound on the region spanned by these parameters. We provide constructive schemes for achieving certain boundary points of this region. Finally, we improve the region by sharing data over aggregated time slots.

1.3.2 ipShield: A Framework For Enforcing Context-Aware Privacy

In Chapter 4, we present a realization of a privacy-aware framework ipShield [CRJ13, CSR14] that builds on the insights provided by the theoretical results. ipShield performs *monitoring* of every sensor used by an app, uses this information to perform a privacy *risk assessment*, provides *recommendation* of possible privacy actions in the form of which sensors to enable and which to disable, and finally,

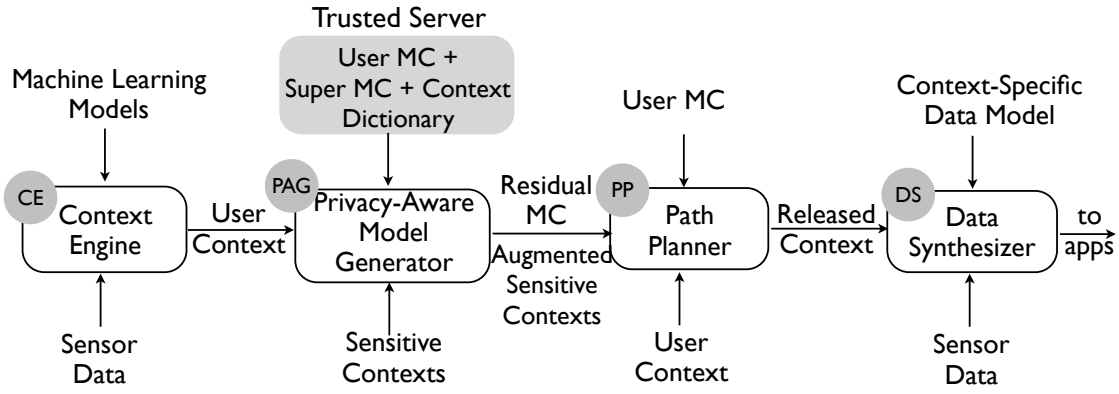


Figure 1.3: iDeceit: A model-based system for plausible falsification of sensor data.

provides with an option to override the generated actions and manually configure context-aware *fine-grained* privacy rules with actions such as data suppression, noise addition and faking of data streams. We implemented ipShield by modifying the internals of the Android operating system (OS) (see Fig. 1.2).

1.3.3 iDeceit: A Framework For Model-Based Data Falsification

We describe our approach towards model-based privacy in Chapter 5. Growing use of location-aware apps that require sharing of user's location necessitate finding methods to protect users' location privacy. Users often visit locations that are sensitive to them and need to be kept private. Existing mitigation techniques include the addition of random noise to the data, which is inadequate because location data often exhibit known temporal and spatial correlation patterns that can be exploited by an adversary to perform denoising. The technique of selective suppression also does not work well as the very act of suppression reveals sensitive information and often apps are not designed to handle data suppression. Finally, manually configuring fine-grained privacy rules may help protect location privacy, but it is cumbersome and hard to remember in every sensitive situation. We present iDeceit (see Fig. 1.3), a framework that implements substitution of sensitive data segments with synthetic data to protect sensitive locations

while ensuring the plausibility of the entire location trace. To ensure plausibility while preserving utility, iDeceit employs a user-behavior model to find candidate segments for substitution that captures the temporal correlation between the different locations visited by the user over time. We show that our model-based substitution ensures zero loss to privacy. The specific model used in iDeceit is a Markov chain (MC).

In addition to comprehensively addressing location privacy, we show that the iDeceit framework is also applicable to other widely used sensor data such as for activity monitoring. To demonstrate real-life usage, we implemented iDeceit on the Android OS by introducing a new data flow path for pushing synthetic sensor data to the apps. We evaluate the efficacy of iDeceit by applying it to protect randomly chosen location data from about 6 million data points collected from a week-long field study with 22 participants. Our evaluation shows that iDeceit protects locations marked sensitive, and its released data (that include substituted segments) are plausible (around 80%) as per user behavior, and it preserves high utility (around 90%), all while incurring zero-loss to privacy.

1.4 Organization of the Document

We start with a review of previous work on privacy and establish the inference privacy problem in Chapter 2. The theoretical aspects of the problem are discussed in Chapter 3. The design and implementation of ipShield is presented in Chapter 4. Model-based privacy is discussed in Chapter 5. We conclude and outline future work in Chapter 6.

CHAPTER 2

Inference Privacy and Review of Prior Work

2.1 Introduction

Smartphones with onboard and externally connected body-worn sensors are used to collect and share personal data with a wide variety of untrusted third-party apps. Information flow from users to apps is summarized in Fig. 2.1. Broadly speaking, the shared sensor data D has: (a) a set of personal identifiers P, such as name and SSN associating it to the user; (b) a set of quasi-identifiers Q, such as age, gender, zip code, which when combined with auxiliary information sources can possibly identify the user; and (c) a set V, containing data values corresponding to the measurement. Shared data is represented by M and is used by the apps to compute various inferences, some of which can be sensitive (i.e., are such that the user wishes them to remain private). Consider the following examples:

E1: A user shares her accelerometer data D with an mHealth app to monitor her overall *activity level and activity type* $f(M)$ but faces the risk of revealing

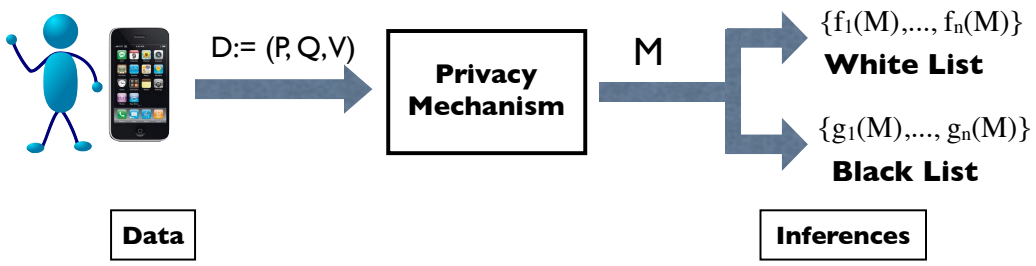


Figure 2.1: Sensor information flow between user and apps.

her *passwords* $g(M)$, which is also inferable from the same data.

E2: A user desiring a *safe driver discount* $f(M)$ on insurance rates may be willing to share her location and accelerometer data D. However, periodic location releases may reveal visits to places leading to sensitive inferences, $g(M)$, such as *religious preferences, drinking habits*, and *health conditions*.

E3: A user is required to share EKG and respiration data D with an insurance company which uses the data to check for *heart and respiratory disorders* $f(M)$, and provide discounted rates. However, the same data can be used to detect *onset of stress* $g(M)$, a behavior the user wants to keep private.

If M is the same as D, the presence of set P implies that the inferred sensitive behaviors may now be traceable back to the user, violating her privacy.

Prior work on privacy mechanisms is centered around two design objectives: data anonymization and incomplete reconstruction. The process of anonymization includes the removal of P and the suitable *obfuscation* of Q present in D to break the association between the data and the user. In a multi-user setting where privacy of an entire database of user data is desired, measures such as *k-anonymity* [Swe02] and *l-diversity* [MKG07] are used to determine the level of obfuscation required to make the user anonymous or indistinguishable within a subpopulation, achieving privacy-in-numbers. However, the breakdown of anonymization in the face of auxiliary information [NS08, GP09, SH12] has prompted the design of measures such as *differential privacy* [Dwo06] which, in a multi-user setting, recommend use of structured noise to perturb aggregate query responses and protect the membership (i.e., presence or absence) of an individual within a database. The second objective is to prevent complete reconstruction of D from M. To achieve this in addition to anonymization, the measurements in V are also adequately perturbed [SRP13]. By preventing reconstruction, the goal is to protect against private inferences which could be made from D alone. However, it

has been shown that partially reconstructed data can be used to make inferences about private behaviors [SH12, GP09].

Now, consider the setting in which a *single* user shares a time-series of sensor data annotated with identity information, as illustrated in examples $E1-E3$. This motivates the question: *what are the privacy and utility goals appropriate to such a setting?* The traditional notion of protecting user identity is no longer a concern because the apps under consideration (e.g., mHealth, customized insurance plans) require user identity for providing personalized services (utility). Thus, instead of identity, a user is interested in protecting the privacy of sensitive behaviors which can be inferred from the shared data. Another consequence of this single-user setting is that privacy measures relying upon privacy-in-numbers within a subpopulation do not apply.

In this chapter, we provide a very general privacy model in which two sets of inferences (the white and black lists shown in Fig. 2.1) constitute utility and private behaviors, respectively. These inferences are marked as $f(M)$ and $g(M)$ in examples $E1 - E3$. The use of inference functions allows us to establish a terminology to unify prior notions of anonymization- and reconstruction-based privacy as special cases of the more general problem. We review prior work in various areas of privacy within the constructs of our model.

We then identify several information disclosure regimes, each corresponding to a specific privacy-utility tradeoff, and indicate privacy mechanisms that can be used to realize these tradeoff points. We describe how the implementation versus privacy consideration at each of the tradeoff points lead us to the design of our privacy system in Chapter 4.

2.2 Inference Privacy Problem

We define an *inference function* as a machine learning algorithm that uses shared data together with a model of the data to make predictions. For example, in $E1$, accelerometer data together with gyroscope data is used to predict the keystroke and hence infer password [MVB12]. In $E2$, proximity or regular visit to a religious place is used to infer religious preferences, visit to a specific hospital could be used to infer medical condition [Kru09]. Finally, in $E3$, respiration data can be used to infer onset of stress [PRH11, LFR12]. These machine learning algorithms use supervised, unsupervised, reinforcement modes to learn a model of data and use it to make predictions about user behavior.

The problem of protecting the privacy of sensitive inferences can be characterized as a tradeoff between the users' need to derive *utility* together with her need to control the information shared for protecting *privacy*. As shown in Fig. 2.1, our privacy notion is defined in terms of what can be extracted from the shared data M. The user specifies her privacy preferences as a *blacklist* of inferences, $\{g_1(M), \ldots, g_n(M)\}$, and the utility requirements as a *whitelist* of inferences, $\{f_1(M), \ldots, f_n(M)\}$. The privacy mechanisms are designed to ensure the app can effectively compute whitelisted inferences to some degree of accuracy, but where the app cannot draw the blacklisted inferences. Ideally, any data shared with an app should not reveal any more information than what is already known to the app about the blacklisted inferences from prior (population-scale) knowledge or side-channels. We remark that this is a general formulation of the privacy problem, and that the previously mentioned privacy mechanisms such as anonymization and protection against reconstruction attacks can be thought of as carefully chosen blacklist inferences.

2.2.1 Information Disclosure Regimes

The various possible tradeoff points for the utility and privacy objectives define a spectrum of information disclosure regimes that a user can operate in. At one extreme, corresponding to zero disclosure, the user shares no information at all, ensuring complete privacy but at the cost of complete loss in utility. At the other extreme, corresponding to full disclosure, all information is shared. In this case, the user achieves full utility at the cost of complete loss of privacy. Other points in this spectrum are realizable by choosing appropriately designed privacy mechanisms. Below we discuss two operating points of particular interest.

1. Maximum Utility Under Perfect Privacy $\max_{\text{perfP}} \text{U}$: We release information (some transformation of D) such that only the desired utility (whitelisted functions, and consequences inferable from them) can be computed from the released information. This point corresponds to targeted disclosure.

2. Maximum Privacy Under Perfect Utility $\max_{\text{perfU}} \text{P}$: We release information which preserve all characteristics of D, except those which can be used to violate privacy (blacklisted functions). This point corresponds to targeted hiding.

We define these terms more precisely in mathematical terms in Chapter 3.

2.3 Prior Work on Privacy Mechanisms

We define a privacy mechanism as a two-step process: first step is to identify the data that is to be shared (e.g., the subset of features, specific samples, inferences). The second step involves applying obfuscation actions on the data before their release. In this section, we summarize various mechanisms that have been used to protect privacy. We indicate how these mechanisms can all be expressed within the framework of whitelisted and blacklisted inference functions.

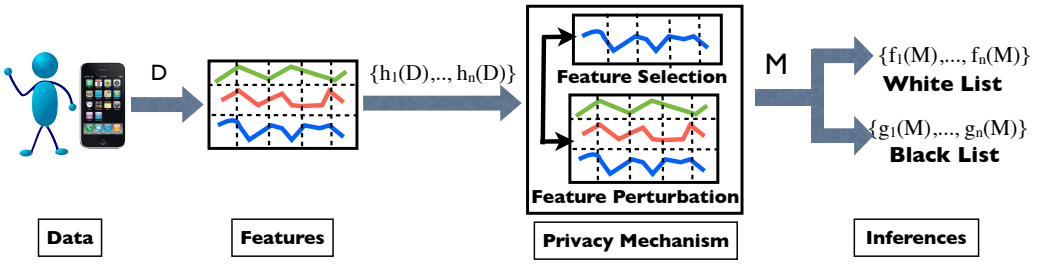


Figure 2.2: Feature selection and perturbation steps for $\max_{\text{perfP}} \text{U}$ and $\max_{\text{perfU}} \text{P}$ respectively.

2.3.1 Feature Selection

Instead of the high-dimensional data D, from which information flow is hard to control [NS10, NS08] we extract a set of features $F = \{h_1(D), \ldots, h_n(D)\}$ and use them to represent the data in a lower-dimensional space. The functions $h_i(D)$ can represent features like mean, variance, Fourier coefficients, etc., extracted from the data samples over time. Inferences typically operate in the feature space and use a subset of F to perform their classification.

To implement $\max_{\text{perfP}} \text{U}$, we observe that by using features we can better control the information shared. The privacy mechanism (see Fig. 2.2) selects a subset of features that are required by the whitelisted inferences but does not contribute to the blacklisted ones [CRS12]. The obfuscation step either suppresses all the other features and shares only the selected features or synthesizes data M preserving only selected features (and their consequences) and nothing else. This mechanism requires the app to specify the features it needs to compute the inference.

2.3.2 Sharing Whitelisted Inferences

Another privacy mechanism which also implements $\max_{\text{perfP}} \text{U}$ is that of sharing whitelisted inferences (or suppressing blacklisted ones). The idea is to compute the inferences on the phone, obfuscate the results such that they do not reveal any information about the blacklisted inferences and share the obfuscated results

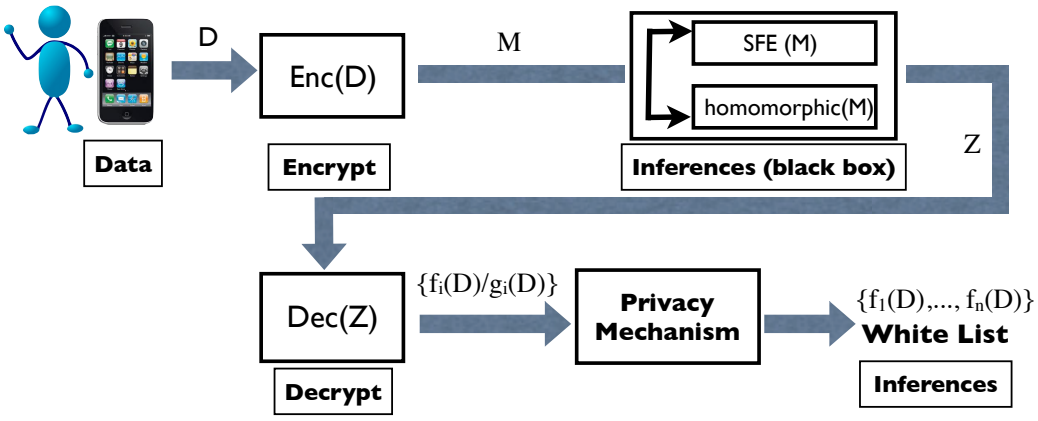


Figure 2.3: SFE and Homomorphic encryption for computation of whitelist functions.

instead of D. For this to work, the apps need to provide the exact implementation of the inference algorithm to the user, which may be proprietary and difficult to share. An alternate strategy for evaluating the whitelisted inference functions is to use cryptographic techniques. We suggest two such techniques (see Fig. 2.3).

- One-sided Secure Function Evaluation (SFE) can applied (using, e.g., Yao's garbled circuit [Yao86]) to evaluating the inference function. Both parties provide their inputs (the user provides her sensor data, and the app the inference function), and the function is evaluated. Since the protocol is one-sided, only the user obtains the result of the computation; and the app knows nothing about the user input. The user can then obfuscate the result before sharing it with the app.

- Homomorphic Encryption [GH11] allows computation to be carried out on the cipher text directly, yielding an encrypted result of the operations performed on the plain text. The user performs homomorphic encryption on the data and sends it to the app, which can then perform function evaluation on the encrypted data and return the encrypted result to the user, who decrypts it to obtain the result. The second step is to perform obfuscation of the result before sharing with the app.

- Functional Encryption: A recent technique proposed in [BSW11] supports restricted secret keys that enable a key holder to learn a specific function of the encrypted data and nothing more about the data. Functional Encryption takes a different approach towards public key encryption. Traditionally, encryption is targeted towards a specific consumer bearing a secret key and the access to the encrypted data is all or nothing - either one can decrypt and read the entire plain text or nothing at all. Instead in functional encryption, the provider does not encrypt for a specific consumer, but only specifies how to share the data. Also a decryption key allows a consumer to learn only a function of the encrypted data.

While the above techniques allow computation of the inference functions without their disclosure, there is no way for the user to know if the results computed are for the whitelisted inferences only. Thus the privacy mechanism must use other techniques (such as zero knowledge proofs [GMR89], random spot checks, etc.), to ensure that the correct functions are being evaluated. In addition, while feasible in theory, these techniques are extremely computationally expensive and thus energy-intensive.

2.3.3 Random Projection

Following this mechanism (see Fig. 2.4), we share projections of the features instead of the features themselves [LKR06]. That is, we project the features into a lower dimensional space before sharing. To ensure that privacy is maintained, the transformation is kept private and is known only to the user.

For utility goals, the user furnishes training labels so that the app can learn a classifier, based on the projected features and associated labels, for the whitelisted inferences (and their consequences) but nothing else. In order to learn the labels in the embedded space, the key property required is that pairwise distances between

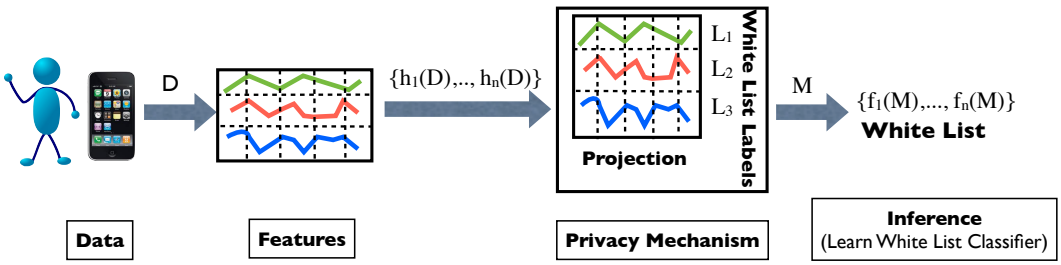


Figure 2.4: Random Projection for computing whitelisted functions. Both projection and the corresponding whitelist labels are shared.

points in the original feature space be preserved. Fortunately, when the transformation is derived from randomly generated basis vectors drawn from an i.i.d. normal distribution, the Johnson Lindenstrauss lemma states that this property holds with high probability when the dimensionality of the new projected feature space satisfies a certain size constraint [KKM12, LKR06].

This mechanism eliminates the need to know a priori the mapping between the inferences and features as required by the feature selection approach. It places a significant burden on the app, however, which must now learn the classifier or the whitelisted inference. An advantage of using this mechanism is that we can guarantee privacy when there is no side-channel information, as only the whitelisted inference labels are shared.

2.3.4 Feature Perturbation

We use this mechanism to realize $\max_{\text{perfU}} \text{P}$ (see Fig. 2.2). We select and transform a specific set of features, and share everything else. For example, for an audio signal we can choose pitch as the feature to transform and use perturbation to obfuscate it. While inferences such as identification of the speaker, which rely on pitch, are affected, other inferences not depending on pitch remain accurately computable. One of the drawbacks of this mechanism is that it does not protect against blacklisted inference functions, which can learn a classifier using the set

of released features instead of the transformed ones.

2.3.5 Differential Privacy

Proposed in [Dwo06] with strong provable privacy guarantees, differential privacy aims to replace the mostly ad-hoc obfuscation strategies with a principled data release mechanism for statistical databases. To achieve differential privacy it is required that the response to a query including or excluding a particular database entry is indistinguishable in the probabilistic sense. This in turn guarantees that an adversary gains negligible information on individual records upon observing the output of a computation regardless of the auxiliary data available to him.

Formally, consider any two databases $D_1, D_2 \in \mathbb{R}^n$ that differ in exactly one entry. Let $\kappa_f(D_1)$ and $\kappa_f(D_2)$ be the responses from D_1 and D_2, respectively, where κ_f is the mechanism used to respond to an arbitrary query $f()$. Randomized mechanism κ_f provides ϵ-differential privacy if

$$\frac{P(\kappa_f(D_1) \in S)}{P(\kappa_f(D_2) \in S)} \leq e^{\epsilon} \tag{2.1}$$

where $S \subseteq Range(\kappa_f)$. The ratio in Eqn. (2.1), represents the "knowledge gain" for an intruder moving from one version of the database to the other. In order to achieve differential privacy, [Dwo06] suggest the use of Laplace based noise addition. However, the calibration of noise magnitude to simultaneously maintain data utility while preserving privacy is an important issue that needs to be addressed for making it practically relevant [SM11].

Predicated on aggregate query/response systems, differential privacy protects against a specific inference, the *membership disclosure* of an entry in a population-scale database. However, behavioral privacy is due to a set of private inferences drawn over individual (not population-scale) data streams shared by providers.

2.4 Discussion

The general form of the privacy problem in which we want to protect a set of behavioral inferences while revealing another set involves a complex interaction of information theory and machine learning. The problem is well-defined only when the blacklisted inferences are not completely contained within the whitelisted inferences. Otherwise, releasing the whitelisted inferences would violate privacy. For well-defined settings, the challenge lies in finding the right subset of features that will enable the whitelisted inferences but have low mutual information with the blacklisted inferences.

CHAPTER 3

Protecting Against Unwanted Inferences

3.1 Introduction

In this chapter, we study the competing goals of utility and privacy preferences of a user in an information theoretic setting. We formulate the goals in terms of the inferences which can be drawn using the shared data. A whitelist describes the inferences that are desirable, i.e., providing utility. A blacklist describes the unwanted inferences which the provider wants to keep private. We formally define utility and privacy parameters using elementary information-theoretic notions, derive a bound on the region spanned by these parameters and provide constructive schemes for achieving certain boundary points of this region.

3.2 The Basic Model

3.2.1 Definitions

In our model (see Fig.3.1), a data provider senses a discrete RV D. The provider cooperates with the recipient by sharing information about the whitelist, as specified by the RV $X = f(D)$ (e.g., X can be mute, ring, or vibrate). The provider also wants to keep the blacklist, specified by $Y = g(D)$ (e.g., Y can be work, home, theater, etc.) private. We consider X and Y to be deterministic functions of D. In this work, we focus on provider strategies for generating message M from D, represented by the distribution $p(M|D)$, such that the desired tradeoff

between utility (M gives a lot of information about X) and privacy (M provides no information about Y) can be achieved.

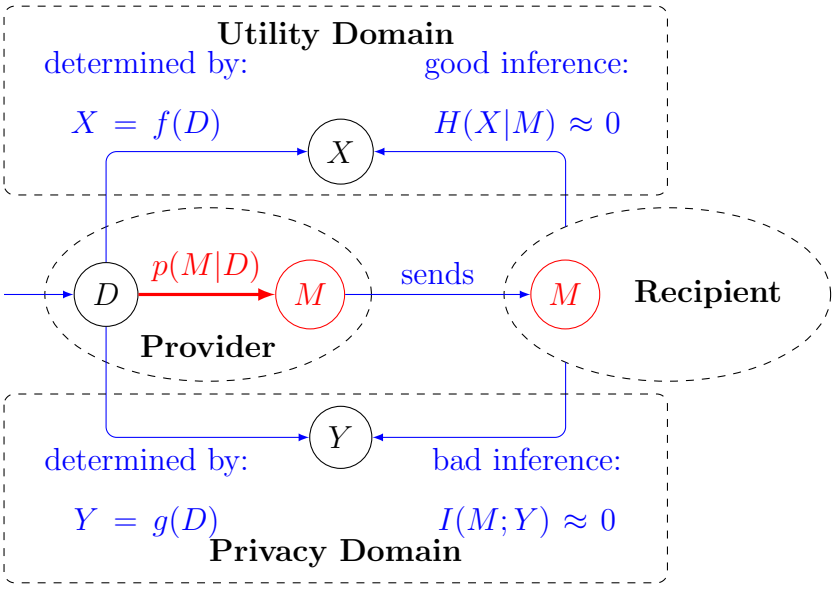


Figure 3.1: The provider senses an RV D and wants to send a message M to the recipient, so that recipient can estimate $X = f(D)$ from M without being able to estimate $Y = g(D)$. If f, g and the distribution of D are fixed, the recipient can only choose the conditional distribution $p(M|D)$.

Formally, for a given choice of distribution $p(M|D)$ of message M knowing data D, we define a utility parameter $\delta_U(M)$ and a privacy parameter $\delta_P(M)$ as:

$$\delta_U(M) \triangleq \frac{H(X|M)}{H(X)}, \quad \delta_P(M) \triangleq \frac{I(M;Y)}{H(Y)}. \tag{3.1}$$

When there is no ambiguity, we omit the argument M of δ_U and δ_P. Depending on the application, various other utility metrics could be useful, for instance Hamming or Euclidean distortions [SRP13]. Most of our analysis directly translates under any distortion metric for utility: for reasons of simplicity we choose to present it here under the equivocation metric δ_U.

These parameters expressed in terms of elementary information-theoretic notions conveniently capture the tradeoff between utility and privacy. The smaller

$\delta_U(M)$ is, the more useful M is in determining X, and the smaller $\delta_P(M)$ is, the more private M is about Y. We refer to $\delta_U = 0$ as the *perfect utility* case (in which $H(X|M) = 0$ and therefore X can be perfectly inferred from M) and to $\delta_P = 0$ as the *perfect privacy* case (in which $I(M;Y) = 0$ and therefore Y is independent from M). In general, it is not possible to achieve perfect utility and perfect privacy at the same time. In the following, we provide a lower bound on the achievable (δ_U, δ_P) pairs:

Theorem 1. *Let D, X and Y be fixed. For any choice of conditional distribution $p(M|D)$, the following lower bound holds:*

$$\delta_U(M)H(X) + \delta_P(M)H(Y) \geq I(X;Y). \tag{3.2}$$

For a given choice of M, equality in (3.2) holds if and only if $H(X|M,Y) = I(M;Y|X) = 0$.

Proof. We prove that for any three RVs M, X and Y, $H(X|M) + I(M;Y) \geq I(X;Y)$, where $\delta_U(M)H(X) = H(X|M)$ and $\delta_P(M)H(Y) = I(M;Y)$:

$$\begin{aligned}
&I(M;Y) + H(X|M) - I(X;Y) \\
&= H(M) + H(Y) - H(M,Y) + H(X|M) - I(X;Y) \\
&= H(M,X) + H(X,Y) - H(M,Y) - H(X) \\
&= H(M) + H(X|M) + H(X) + H(Y|X) \\
&\qquad\qquad - H(M) + H(Y|M) - H(X) \\
&= H(X|M) + H(Y|X) - H(Y|M) \\
&= H(X|M) + I(M;Y|X) + H(Y|M,X) - H(Y|M) \\
&= H(X|M,Y) + I(M;Y|X) \geq 0.
\end{aligned} \tag{3.3}$$

Because $H(X|M,Y)$ and $I(M;Y|X)$ are both non-negative, equality holds if and only if they are both zero. □

From Theorem 1, we deduce that it is possible to achieve perfect utility and perfect privacy at the same time only if $I(X;Y) = 0$ (in fact, one can observe that reciprocally, if $I(X;Y) = 0$ then we can always achieve perfect utility and perfect privacy simultaneously, e.g. by transmitting $M = X$).

A more intriguing question is, when $I(X;Y) > 0$, what are the achievable pairs (δ_U, δ_P)? In this case, we consider that D, X and Y are fixed, and we want to find RVs M that achieve good tradeoffs between δ_U and δ_P. We denote the respective alphabets of D, X, Y and M by $\mathcal{D}$, $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{M}$. While the former three alphabets are fixed by the problem setup, we have the choice of the alphabet $\mathcal{M}$ of M.

To answer the posed question, we first define the *maximum privacy under perfect utility* $\max_{\text{perfU}} \text{P}$ and *maximum utility under perfect privacy* $\max_{\text{perfP}} \text{U}$ points by

$$\begin{aligned} \max_{\text{perfU}} \text{P} &= (0, \delta_P^*) \text{ where } \delta_P^* \stackrel{\Delta}{=} \min_{\substack{M: \\ \delta_U(M)=0}} \delta_P(M), \\ \max_{\text{perfP}} \text{U} &= (\delta_U^*, 0) \text{ where } \delta_U^* \stackrel{\Delta}{=} \min_{\substack{M: \\ \delta_P(M)=0}} \delta_U(M). \end{aligned} \tag{3.4}$$

In the next sections, we consider strategies for points $\max_{\text{perfU}} \text{P}$ and $\max_{\text{perfP}} \text{U}$.

3.2.2 Maximizing Privacy under Perfect Utility

Using the fact that X is a deterministic function of D, we provide a simple strategy to achieve $\max_{\text{perfU}} \text{P}$, and show that the strategy reaches the bound from Theorem 1.

Lemma 2. *For fixed D, X and Y, sharing $M = X$ achieves* $\max_{\text{perfU}} \text{P}$ *and* $\delta_P^* = \frac{I(X;Y)}{H(Y)}$.

Proof. Follows directly from the definition of $\delta_U(M)$ and $\delta_P(M)$ in (3.1) and the bound in Theorem 1. □

Sharing $M = X$ is not useful when the computation of X by the recipient is the only reason the provider agrees to release data. However, it is a valid strategy

when the provider has additional incentive to share data, or when X is only an intermediate variable which requires further processing by the recipient.

3.2.3 Maximizing Utility under Perfect Privacy

As shown in Section 3.2.2, achieving $\max_{\text{perfU}} \text{P}$ is straightforward as it does not depend on Y. However, achieving $\max_{\text{perfP}} \text{U}$ is more involved as it depends on both X and Y. This section aims at providing necessary conditions on $p(D, M)$ for M to achieve point $\max_{\text{perfP}} \text{U}$. Combining these conditions greatly restricts the space over which distributions $p(D, M)$ may achieve $\max_{\text{perfP}} \text{U}$; in fact, it allows us to express the maximization of utility under perfect privacy as a linear programming problem.

3.2.3.1 Necessary Conditions on $p(D, M)$ for Maximum Utility under Perfect Privacy

We provide three conditions. The first one is a necessary and sufficient condition for perfect privacy (with no constraint on utility). It is a simple consequence of our definition of perfect privacy, but we state it formally as it is convenient for our analysis. The remaining two are necessary conditions for maximum utility.

Condition 1. *For all $m \in \mathcal{M}$ and $y \in \mathcal{Y}$,*

$$\sum_{d \in g^{-1}(y)} p(d, m) = p(m)p(y). \tag{3.5}$$

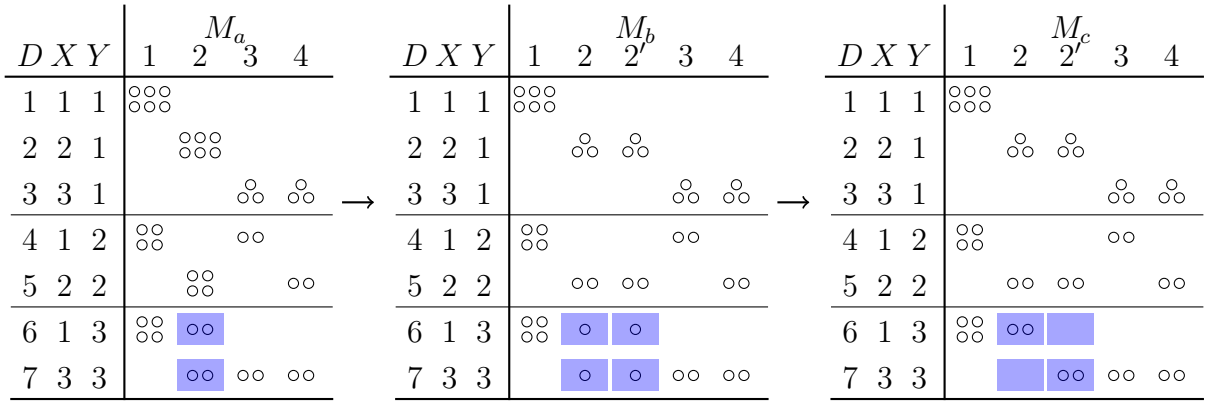
Lemma 3. *RV M achieves perfect privacy if and only if M meets Condition 1.*

Proof. By definition of perfect privacy, $I(M; Y) = 0$ and for all $m \in \mathcal{M}$ and $y \in \mathcal{Y}$ we have $p(m, y) = p(m)p(y)$. The statement follows by noting that

$$p(m, y) = \sum_{d \in g^{-1}(y)} p(d, m). \tag{3.6}$$

□

We now try to maximize utility when Condition 1 is satisfied. Suppose that for every $m \in \mathcal{M}$ there is a unique $x \in \mathcal{X}$ such that $p(m, x) > 0$. Then, M achieves



(a) While Condition 2 is met, Condition 3 is not: we split column $m = 2$ into 2 and $2'$, and obtain M_b that violates Condition 2.

(b) While Condition 3 is now met, Condition 2 is not. We rearrange the highlighted probabilities and gain utility without losing privacy.

(c) Lastly, Conditions 1–3 are met, the table is a candidate for maximum utility under perfect privacy (and does in fact achieve it).

Figure 3.2: Illustration of the impact of Conditions 2 and 3 on the joint distributions $p(D, M)$, when M already meets Condition 1. Each $\circ$ represents a measure of $\frac{1}{49}$. Starting from M_a (left panel) which does not meet Condition 3, we reach M_b (center panel) with $\delta_U(M_b) = \delta_U(M_a)$ and $\delta_P(M_b) = \delta_P(M_a) = 0$, where M_b does not respect Condition 2. From M_b, we reach M_c (right panel) which still has $\delta_P(M_c) = 0$, and has $\delta_U(M_c) < \delta_U(M_b)$.

perfect utility. This observation leads to the intuition that utility is increased when each value $m \in \mathcal{M}$ jointly occurs only with a limited number of different values of $x \in \mathcal{X}$. Therefore, starting from a given RV M that achieves perfect privacy, we might be able to improve utility by performing local rearrangements of the joint distribution $p(D, M)$. The goal is to reduce the number of different values of X that can jointly occur with each value of M while preserving perfect privacy. We identify local patterns in the joint distribution $p(D, M)$ that can be manipulated for a guaranteed increase in utility at no cost in terms of privacy: if for a given M, $p(D, M)$ shows one of these patterns, then M does not achieve $\max_{\text{perfP}} \mathrm{U}$.

Condition 2. *Given $d_1 \neq d_2$ in $\mathcal{D}$ and $m_1 \neq m_2$ in $\mathcal{M}$ such that $f(d_1) \neq f(d_2)$*

and $g(d_1) = g(d_2)$*, there exists a pair* $(i, j) \in \{1, 2\}^2$ *such that* $p(d_i, m_j) = 0$*.*

Lemma 4. *If* M *achieves* $\max_{\text{perfP}} \text{U}$*, then* M *meets Condition 2.*

Proof. Suppose that M achieves $\max_{\text{perfP}} \text{U}$ but does not meet Condition 2. Then, there exist d_1, d_2, m_1 and m_2 so that all of the $p(d_i, m_j)$ terms, $(i, j) \in \{1, 2\}^2$, are non-zero. We show that this leads to a contradiction by building an RV M_α which also achieves perfect privacy, but with better utility than M.

For a real number α in a well-chosen range, we define the RV M_α by its joint distribution $p_\alpha(D, M)$:

$$p_\alpha(d, m) = \begin{cases} p(d, m) + \alpha & \text{if } (d = d_1 \wedge m = m_1) \\ & \text{or } (d = d_2 \wedge m = m_2), \\ p(d, m) - \alpha & \text{if } (d = d_1 \wedge m = m_2) \\ & \text{or } (d = d_2 \wedge m = m_1), \\ p(d, m) & \text{otherwise.} \end{cases} \tag{3.7}$$

Notice that the joint distribution $p_\alpha(D, M_\alpha)$ is a locally perturbed version of $p(D, M)$. We now consider how $H(X|M_\alpha)$ varies with α. Because $p_\alpha(d, m)$ must remain between 0 and 1, α can only vary in some line segment $[\alpha_{\min}, \alpha_{\max}]$. Note that the constraints that the probabilities must remain non-negative are sufficient to find the bounds: taking for instance $p(d_1, m_1) + \alpha > 1$ would imply that $p(d_1, m_2) - \alpha < 0$. Therefore, $\alpha_{\min} = -\min(p(d_1, m_1), p(d_2, m_2))$ and $\alpha_{\max} = \min(p(d_2, m_1), p(d_1, m_2))$.

We now prove that $H(X|M_\alpha)$ is a concave function of α and therefore reaches its minimum for $\alpha = \alpha_{\min}$ or $\alpha = \alpha_{\max}$ (so that one of the $p_\alpha(d_i, m_j) = 0$). We use the following notation shortcuts

$$p^{ij} \triangleq p(m_j, x_i) \; = \sum_{d \in f^{-1}(x_i)} p(d, m_j), \tag{3.8}$$

$$p_\alpha^{ij} \triangleq p_\alpha(m_j, x_i) = \sum_{d \in f^{-1}(x_i)} p_\alpha(d, m_j), \tag{3.9}$$

where $x_i = f(d_i)$, and decompose $H(X|M_\alpha)$ into a term that depends on α and a constant. We use the fact that for all m and all x, $p_\alpha(m) = p(m)$ and $p_\alpha(x) = p(x)$. After some lengthy derivations we obtain

$$\begin{aligned} &H(X|M_\alpha) = \\ &- p_\alpha^{11} \log p_\alpha^{11} - p_\alpha^{12} \log p_\alpha^{12} - p_\alpha^{21} \log p_\alpha^{21} - p_\alpha^{22} \log p_\alpha^{22} + C. \end{aligned} \tag{3.10}$$

Differentiating twice, we get

$$\frac{\partial^2}{\partial \alpha^2} H(X|M_\alpha) = -\frac{1}{p_\alpha^{11}} - \frac{1}{p_\alpha^{12}} - \frac{1}{p_\alpha^{21}} - \frac{1}{p_\alpha^{22}} < 0. \tag{3.11}$$

Thus, there exists an M_α such that $\delta_U(M_\alpha) < \delta_U(M)$.

Also, $p_\alpha(m, y) = \sum_{d \in g^{-1}(y)} p_\alpha(d, m)$, thus for $m \notin \{m_1, m_2\}$ or for $y \neq g(d_1) = g(d_2)$, we have $p_\alpha(m, y) = p(m, y)$ because for these values of d and m, $p_\alpha(d, m) = p(d, m)$. For $m \in \{m_1, m_2\}$ and $y = g(d_1) = g(d_2)$, we have

$$p_\alpha(m, y) = \sum_{d \in g^{-1}(y)} p_\alpha(d, m) = \sum_{d \in g^{-1}(y)} p(d, m) + \alpha - \alpha = p(m, y). \tag{3.12}$$

Therefore, $\delta_P(M_\alpha) = \delta_P(M) = 0$, which together with $\delta_U(M_\alpha) < \delta_U(M)$ contradicts the fact that M achieves $\max_{\text{perfP}} \text{U}$. □

Condition 3. *For all $m_0 \in \mathcal{M}$ and $d_1, d_2 \in \mathcal{D}$ such that $g(d_1) = g(d_2)$,*

$$d_1 \neq d_2 \Rightarrow p(d_1, m_0) = 0 \vee p(d_2, m_0) = 0. \tag{3.13}$$

Lemma 5. *If M achieves $\max_{\text{perfP}} \text{U}$, then M meets Condition 3.*

Proof. Suppose that M achieves $\max_{\text{perfP}} \text{U}$ but does not meet Condition 3. Then, there exist m_0, d_1 and d_2 such that $g(d_1) = g(d_2)$ and both $p(d_1, m_0) > 0$ and $p(d_2, m_0) > 0$. Now consider the RV M' on alphabet $\mathcal{M}' = \mathcal{M} \cup \{m_0'\}$ (where $m_0' \notin \mathcal{M}$ is a new symbol), obtained by splitting symbol m_0 into m_0 and m_0'. Formally,

$$p'(d, m) = \begin{cases} p(d, m) & \text{if } m \notin \{m_0, m_0'\}, \\ p(d, m)/2 & \text{if } m \in \{m_0, m_0'\}. \end{cases} \tag{3.14}$$

Let us first notice that $\delta_U(M') = \delta_U(M)$:

$$\begin{aligned} H(X)\delta_U(M') &= \sum_{m \in \mathcal{M}'} p'(m) H(X|M'=m) \\ &= \sum_{m \in \mathcal{M} \setminus \{m_0\}} p(m) H(X|M=m) + 2\frac{p(m_0)}{2} H(X|M=m_0) \\ &= H(X)\delta_U(M). \end{aligned} \tag{3.15}$$

Similarly, $\delta_P(M') = \delta_P(M) = 0$. As M' does not satisfy Condition 2 (for m_0, m_0', d_1 and d_2), by Lemma 4, M' does not achieve $\max_{\text{perfP}} \text{U}$, and so neither does M. □

Under the assumption that Condition 1 is satisfied, Fig. 3.2 illustrates for a simple synthetic example of D, X and Y how better utility can be obtained once Conditions 2 and 3 are met.

3.2.3.2 Linear Programming Approach for Maximum Utility under Perfect Privacy

We now use Conditions 1–3 to formulate the maximization of utility under perfect privacy as a LP problem to compute $p(D, M)$ so that M achieves $\max_{\text{perfP}} \text{U}$.

We start by defining the *support of D given $M = m$* as the set $S(m) = \{d \in \mathcal{D} : p(d, m) > 0\}$. Condition 3 can be written in terms of S as follows:

$$\forall m \in \mathcal{M}, \forall y \in \mathcal{Y}, |S(m) \cap g^{-1}(y)| = 1. \tag{3.16}$$

Without loss of generality, we only consider those M's for which no distinct m_1 and m_2 have the same support. Otherwise, m_1 and m_2 can be merged into a single symbol with no effect on δ_U and δ_P (for the same reason for which we could split m_0 into two symbols in the proof of Lemma 5).

We denote by $\mathcal{S}$ the set of all possible supports S so that (3.16) is met. For the same instance of D, X and Y as in Fig. 3.2, we show in Fig. 3.3 the structure of the

			M										
D	X	Y	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	a_1	a_2	a_3	a_4	0	0	0	0	0	0	0	0
2	2	1	0	0	0	0	a_5	a_6	a_7	a_8	0	0	0	0
3	3	1	0	0	0	0	0	0	0	0	a_9	a_{10}	a_{11}	a_{12}
4	1	2	b_1	b_2	0	0	b_5	b_6	0	0	b_9	b_{10}	0	0
5	2	2	0	0	b_3	b_4	0	0	b_7	b_8	0	0	b_{11}	b_{12}
6	1	3	c_1	0	c_3	0	c_5	0	c_7	0	c_9	0	c_{11}	0
7	3	3	0	c_2	0	c_4	0	c_6	0	c_8	0	c_{10}	0	c_{12}

Figure 3.3: General structure of the joint distribution for M satisfying Conditions 1–3 (for an example choice of X and Y). For each $m \in \{1, \ldots, 12\}$, $\frac{a_m}{3} = \frac{b_m}{2} = \frac{c_m}{2}$. The table is thus entirely determined by the choice of the values $p(m) = a_m + b_m + c_m$ for each $m \in \{1, \ldots, 12\}$.

candidate joint distributions for $\max_{\text{perfP}} \text{U}$. Notice that for any given m, Condition 1 requires that $\frac{a_m}{3} = \frac{b_m}{2} = \frac{c_m}{2}$, which in turn determines the conditional distribution of Y given $M = m$. Also, $|\mathcal{S}| = \prod_{y \in \mathcal{Y}} |g^{-1}(y)| = 3 \times 2 \times 2 = 12$. For M achieving Conditions 1–3, we can therefore characterize the joint distribution $p(D, M)$ with only $|\mathcal{S}|$ values (the vector $(p(m))_{m \in \mathcal{M}}$ with $\mathcal{M} = \{1, \ldots, |\mathcal{S}|\}$). This greatly reduces the dimensionality of the space of the candidate joint distributions for $\max_{\text{perfP}} \text{U}$.

We therefore formulate achieving $\max_{\text{perfP}} \text{U}$ as the problem of finding $(p(m))_{m \in \mathcal{M}} \in [0, 1]^{|\mathcal{M}|}$ which minimizes

$$H(X|M) = \sum_{m \in \mathcal{M}} p(m) H(X|M = m), \tag{3.17}$$

(where $\mathcal{M}$ is chosen so that $\{S(m) : m \in \mathcal{M}\} = \mathcal{S}$ with each support represented exactly once), under the constraints that for each $d \in \mathcal{D}$,

$$\sum_{m : d \in S(m)} p(d, m) = p(d), \tag{3.18}$$

which can be written in terms of the $p(m)$:

$$\sum_{m : d \in S(m)} p(m) \Pr(Y = g(d)) = p(d). \tag{3.19}$$

The above LP has an average-case complexity which is polynomial in $|\mathcal{S}|$, where $|\mathcal{S}|$ is upper bounded by $\left(\frac{|\mathcal{D}|}{|\mathcal{Y}|}\right)^{|\mathcal{Y}|}$.

3.3 Model Extension

3.3.1 Obtaining Better Results by Grouping Samples

While $\max_{\text{perfU}} \text{P}$ always achieves the bound from Theorem 1 as discussed in Section 3.2.2, $\max_{\text{perfP}} \text{U}$ in general does not. We try to reduce the gap between $\max_{\text{perfP}} \text{U}$ and the theoretical bound by simultaneously using multiple time-slots. Rather than considering three RVs D, X and Y, we consider that they are part of three i.i.d. random processes (D_t), (X_t) and (Y_t). If we send a message M_t independently at every time slot t, the analysis remains the same as before. Hoping to reach better privacy and utility, we decompose the processes into groups of T time slots and treat each of these groups as a whole.

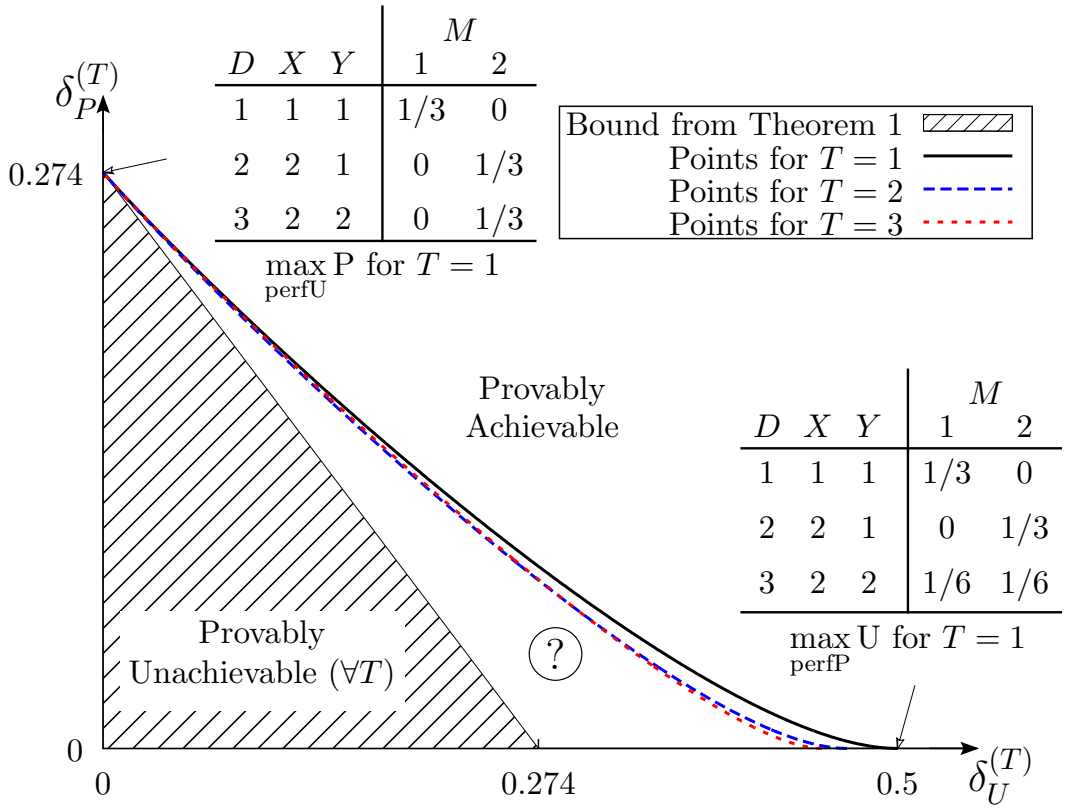
For an RV $M^{(T)}$, we define the utility and privacy parameters corresponding to a group of T time slots:

$$\delta_U^{(T)}(M^{(T)}) \triangleq \frac{H(X_1^T|M^{(T)})}{H(X_1^T)}, \delta_P^{(T)}(M^{(T)}) \triangleq \frac{I(M^{(T)};Y_1^T)}{H(Y_1^T)}. \tag{3.20}$$

While $M^{(T)}$ plays the same role as M did before, we include a superscript (T), to indicate that M spans T time slots. We may omit this superscript when $T = 1$. The following lemma provides a way of obtaining RVs $M^{(T)}$ using an RV M designed for a single time slot.

Lemma 6. *Let f and g be two (deterministic) functions, and (D_t), (X_t) and (Y_t) be three i.i.d. random processes so that for each t, $X_t = f(D_t)$ and $Y_t = g(D_t)$. Then, for any RV M given by the joint distribution $p(D, M)$, consider the i.i.d. random vector M_1^T over $\mathcal{M}^T$ given by its joint distribution with D_1^T,*

$$p^{(T)}(d_1^T, m_1^T) = \prod_{t=1}^{T} p(d_t, m_t). \tag{3.21}$$



D	X	Y	$M=1$	$M=2$
1	1	1	1/3	0
2	2	1	0	1/3
3	2	2	0	1/3

D	X	Y	$M=1$	$M=2$
1	1	1	1/3	0
2	2	1	0	1/3
3	2	2	1/6	1/6

Figure 3.4: Points $(\delta_U^{(T)}, \delta_P^{(T)})$ for $\max_{\text{perfU}} \text{P}$, $\max_{\text{perfP}} \text{U}$ and tradeoffs between the two, for T from 1 to 3, on an example with $|\mathcal{D}| = 3$. While $\max_{\text{perfU}} \text{P}$ does not change with an increase in T, $\max_{\text{perfP}} \text{U}$ is shifted leftwards, and better tradeoff points are achieved for $T = 2, 3$ than for $T = 1$.

Here, M_1^T conserves the utility and privacy parameters of M:

$$\delta_U^{(T)}(M_1^T) = \delta_U(M), \ \delta_P^{(T)}(M_1^T) = \delta_P(M). \tag{3.22}$$

Proof. The proof follows from elementary information-theoretic properties of i.i.d. processes. □

Lemma 6 ensures that the optimal $(\delta_U^{(T)}, \delta_P^{(T)})$ pairs are no worse than the optimal (δ_U, δ_P) pairs. It is in fact possible to build RVs for T time slots that achieve results strictly better than the best that can be obtained for a single time slot. For instance, in Fig. 3.4, we consider a small example with $\mathcal{D} = \{1, 2, 3\}$, $f(1) = f(2) = g(1) = 1$, $f(3) = g(2) = g(3) = 2$. We plot the bound from Theorem 1, which delimits a provably unachievable region. For each $T \in \{1, 2, 3\}$, we

also plot the points $\max_{\text{perfU}} \text{P}$ and $\max_{\text{perfP}} \text{U}$ obtained using respectively Sections 3.2.2 and 3.2.3, and use a heuristic algorithm to achieve and plot tradeoffs between utility and privacy by combining the joint distributions of $\max_{\text{perfU}} \text{P}$ and $\max_{\text{perfP}} \text{U}$. Combining these two distributions requires computing "compatibility scores" between elements of the alphabets for the extreme points and using these scores to carefully construct a common alphabet. The detailed description of this procedure is left for an extended version of the paper. For each T, these tradeoffs form a curve that delimits a region of provably achievable pairs (δ_U, δ_P). The maximum utility that can be reached under perfect privacy using a single time slot is $\delta_U^{(1)*} = 0.5$. However, with two time slots, it can be reduced to $\delta_U^{(2)*} = 0.468$, and to $\delta_U^{(3)*} = 0.452$ for three time slots. It appears challenging to establish whether the points in the area in between the theoretical bound and the heuristically obtained tradeoffs are achievable or not (except for the points $(\delta_U^{(T)}, 0)$ with $\delta_U^{(T)} < \delta_U^{(T)*}$, which are unachievable by definition).

The bound from Theorem 1 remains the same regardless of T because of the assumption that the processes are i.i.d.: for any T, $\delta_U^{(T)*} \geq \frac{I(X;Y)}{H(X)} = 0.274$. An interesting question is to determine if $\delta_U^{(T)*}$ approaches this bound when T goes to infinity. The time complexity of the method from Section 3.2.3.2 prohibits the computation of $\max_{\text{perfP}} \text{U}$ for $T > 3$, even for examples like the one in Fig. 3.4.

3.3.2 Performance with Imperfect Knowledge of the Joint Distribution

Until now, we considered that the process of determining a good masking protocol was conducted with perfect knowledge of the distribution of D. However, in more practical scenarios, one would probably only have access to an estimate of that distribution by sampling it. It is therefore relevant to wonder how much the (δ_U, δ_P) for critical point such as $\max_{\text{perfP}} \text{U}$ and $\max_{\text{perfU}} \text{P}$ evolve when our knowledge of the distribution of D only comes from sampling it.

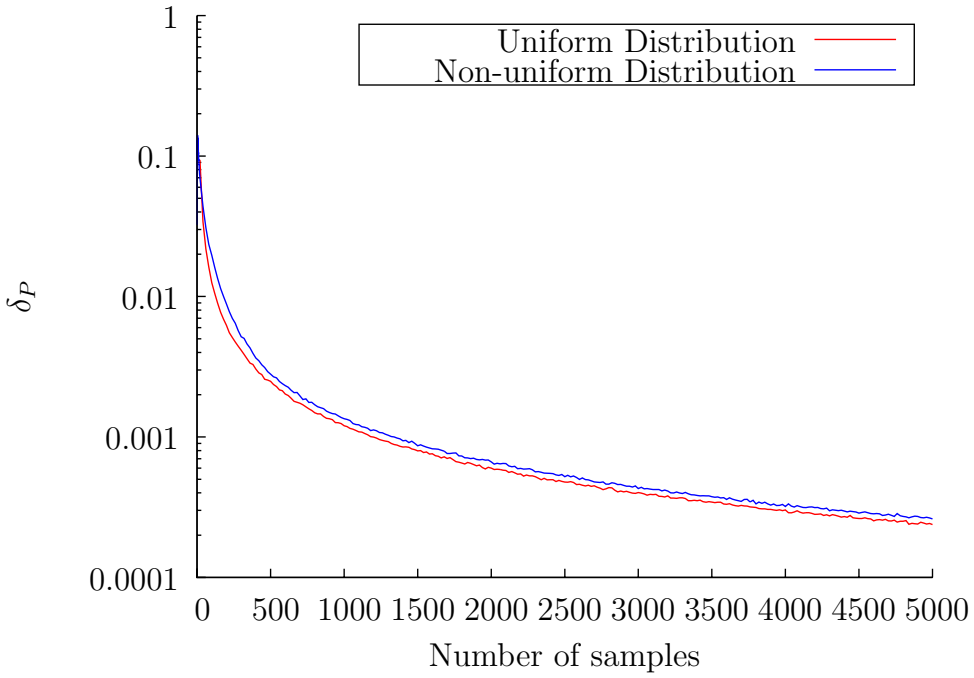


Figure 3.5: Privacy loss δ_P resulting by using a sampled distribution of D to determine $\max_{\text{perfP}} \text{U}$, compared to the ideal case in which perfect knowledge of the distribution of D. The experiment is conducted for two distributions of D over an alphabet of size 7: the uniform distribution and a non-uniform one.

For $\max_{\text{perfU}} \text{P}$, since our strategy is to simply transmit $M = X$, as long as we have perfect knowledge of the (deterministic) function f that relates X to D, the masking protocol does not depending on the distribution of D and is therefore unaffected by an imperfect knowledge of that distribution. However, this is not the case for $\max_{\text{perfP}} \text{U}$.

In the following, we consider the same example choice of D, X and Y as in Fig. 3.2 under two distributions of D: uniform distribution over $\{1, \ldots, 7\}$ and the distribution given by the vector $[0.1, 0.05, 0.4, 0.01, 0.1, 0.1, 0.24]$ (referred to as "Non-uniform distribution"). For both, we apply the linear programming approach from Section 3.2.3 to compute the $\max_{\text{perfP}} \text{U}$ masking strategy, but instead of using the distributions themselves for the computations, we sample them and

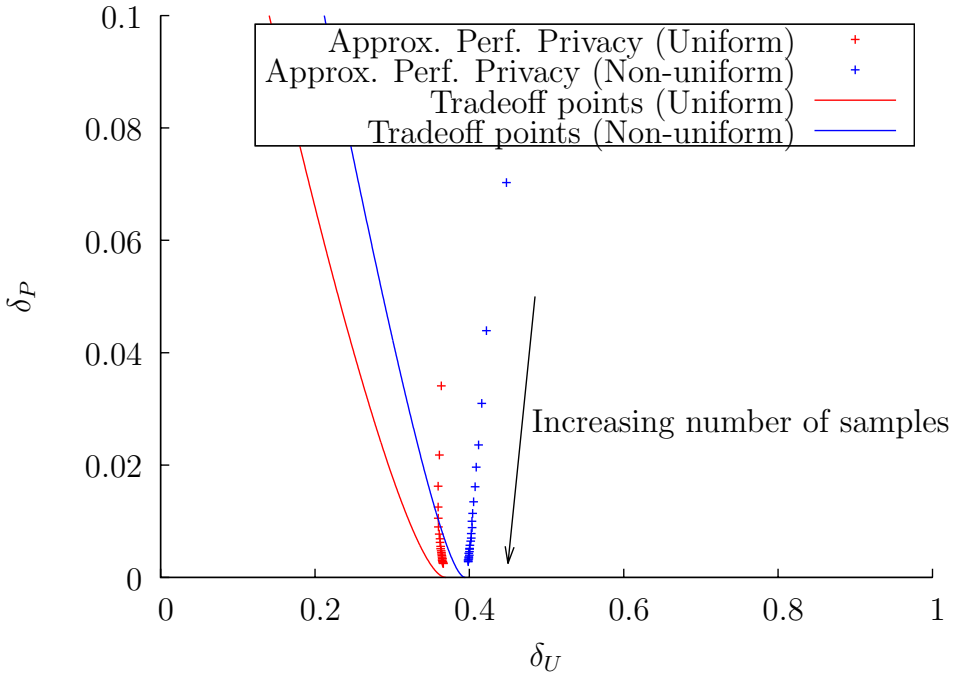


Figure 3.6: Pairs (δ_U, δ_P) obtained by sampling a distribution $n = 20, 40, \ldots, 500$ times and using that empirical distribution to determine $\max_{\text{perfP}} \text{U}$. The experiment is conducted for two distributions of D over an alphabet of size 7: the uniform distribution and a non-uniform one.

use the resulting empirical distributions instead.

Fig. 3.5 shows how the sample size affects the δ_P parameter of the protocol generated with this imperfect knowledge, for n up to 5000. For each value of n, we averaged δ_P over a thousand draws of a sample distribution of size n. We observe that for both original distribution of D, as few as 200 samples are sufficient to reach a δ_P below one percent. We conclude that our method is resilient to imperfect knowledge of the distribution of D and can therefore be used on empirically determined distributions at almost no performance cost.

δ_U is in average mostly unaffected. Fig. 3.6 reports the (δ_U, δ_P) pairs obtained by sampling the same two distributions as in Fig. 3.5 over 20, 40, ..., 500 samples, as well as the tradeoff curves derived with perfect knowledge of the distributions as references (points closer to $\delta_P = 0$ correspond to more samples). There is

a very slight gain in utility in the uniform case and a very slight loss in the non-uniform case, but both are negligible, and futher decrease as the number of samples increases.

3.4 Discussion

In this chapter, we present both theoretical and practical results with the goal of improving user privacy without preventing legitimate inferences from being made.

Our theoretical framework allows the user to specify the inferences to protect and those to allow. Universal metrics are defined to quantify the utility and privacy-leakage that a given masking protocol provides. Application-dependent metrics (e.g., suitable distortions) are also compatible with our model. We provide schemes that maximize privacy under a perfect utility constraint and vice versa. We propose methods to reach heuristic tradeoffs in between these two extreme scenarios, and derive theoretical bounds to the achievable region of the utility/privacy tradeoffs.

CHAPTER 4

ipShield: A Framework For Enforcing Context-Aware Privacy

4.1 Introduction

Smartphones have evolved from mere communication devices into sensing platforms supporting a sprawling ecosystem of apps which thrive on the continuous and unobtrusive collection of personal sensory data. This data is often used by the apps to draw inferences about our personal, social, work and even physiological spaces [sag, LPL09, BI04, RMB10, run, ZCC12, PRH11, RAP11] often under the pretext of providing personalized experiences and customized recommendations. However, not all app developers are equally trustworthy, and this coupled with user naïveté leads to data misuse and privacy concerns.

To safeguard user privacy, Android requires developers to specify the permissions needed by their apps. At install time, the user can either grant access to all the requested resources or opt to not use the app at all. But despite these provisions, cases of privacy violations by third-party apps are rampant [HHJ11, pau, TK]. We observe multiple problems with the current privacy mechanism in Android. First, only a select set of sensors such as GPS, camera, bluetooth are considered to be privacy-prone and have their access mediated through protected APIs [sec]. Other onboard sensors such as accelerometer, gyroscope, light, etc. are considered to be innocuous, requiring no user permission. This specific vulnerability of unrestricted access to accelerometer and gyroscope

data has been exploited to mount keylogging attacks [MVB12], and reconstruction of travel trajectories [HON12]. Second, various studies [RGK11, FHE12] to understand users' perception of privacy in general and their understanding of Android permissions in particular reveal that users are often oblivious to the implications of granting access to a particular type of sensor or resource on their phone at install time. However, the perception quickly changes to one of concern when apprised of the various sensitive inferences that could be drawn using the shared data. Finally, users only have a binary choice of either accepting all the requested permissions or not installing the app at all. Once installed, users do not have any provision to revoke or modify the access restrictions during runtime.

Prior research has tried to address some of the above problems. TaintDroid [EGC10], extends the Android OS by adding taint bits to sensitive information and tracking the flow of those bits through third-party apps to detect malicious behavior. However, tainting sensor data continuously for all apps has high runtime overhead, and is often conservative as data sensitivity typically depends on user context. Moreover, TaintDroid stops at detection and does not provide any recommendation on countering the privacy threat. MockDroid [BRS11], is a modified Android OS designed to allow users the ability to mock resources requested by the app at runtime. Mocking is used to simulate the absence of resources (e.g., lack of GPS fix, or Internet connectivity), or provide fixed data. However, MockDroid only works for resources explicitly requested by an app (no innocuous sensors), is binary because a user can either mock a resource or provide full access to it and finally MockDroid falls short on providing any guidance to the user regarding which sensors to mock. PMP [AH13], is a system that runs on iOS and allows users to control access to resources at runtime. It uses a crowdsourced recommendation engine to guide users towards effective privacy policies. However, PMP does not handle sensor data.

In this chapter, we present ipShield [ipS], a privacy-enforcing framework on the

Android OS. ipShield allows users to specify their privacy preferences in terms of semantically-meaningful inferences that can be drawn from the shared data and, if required, also configure fine-grained privacy rules for every accessed sensor on a per app basis at runtime. We build on prior work in [pdr, BRS11] and make the following contributions.

- We modify the Android OS to monitor all the sensors accessed by an app regardless of whether they are specified explicitly by the app at install time. As per our knowledge, ours is the first system that tracks innocuous sensors.
- We take an important step towards presenting the privacy risks in a more user-understandable format. Instead of listing sensors, we list the inferences that could be made using the accessed sensors. Users can specify their privacy preferences in the form of a prioritized blacklist of private inferences and a prioritized whitelist of allowed inferences.
- We implemented a recommendation engine to translate the blacklist and the whitelist of inferences into lower-level privacy actions (suppression, allow) on individual sensors.
- Finally, we provide the user with options to configure context-aware fine-grained privacy actions on different sensors on a per app basis at runtime. These actions range in complexity from simple suppression to setting constant values, adding noise of varying magnitude, and even play-back of synthetic sensor data.

ipShield is open source and implemented by modifying Android Open Source (AOSP) [aos] version $4.2.2\_r1$. We evaluated it using computation intensive apps requiring continuous sensor data. Our results indicate that ipShield has negligible CPU and memory overhead and the reduction in battery life is around $\sim 8\%$.

4.2 Case Studies

Using two typical scenarios we illustrate below how ipShield would help app users protect their privacy.

4.2.1 Transportation Mode and KeyLogging

Activity recognition algorithms [BI04, RMB10] are used by various fitness and wellness apps to infer the users' Transportation Mode (e.g., predict one of three labels: *walking*, *motorized* or *still*). For example, the Ambulation app in [RMB10] combines accelerometer and GPS data to infer the labels with over 90% accuracy. However, data collected for inferring Transportation Mode can also be used to infer other labels sensitive to the user. For example, the same accelerometer when combined with gyroscope data can be used to infer Onscreen Taps and capture keystrokes on the softkeyboard [MVB12] (and also Location [HON12]) with over 80% accuracy. This could lead to leak of sensitive information like password and PIN entered on the phone.

Using ipShield, a user would add the Transportation Mode and the Onscreen Taps to the whitelist and the blacklist, respectively. This will block the accelerometer and gyroscope data from reaching the Ambulation app preventing keylogging. However, this will also cause the app to stop performing activity recognition. In Section 4.8 we show how ipShield allows users to configure fine-grained rules to maximize the utility of the app.

4.2.2 Saga: Location

Saga [sag] is a life logging app which runs in the background and keeps track of the places a user has visited. By analyzing the location trace of a user, Saga can infer useful information such as average daily commute time, time spent at work, etc. However, it can also derive sensitive inferences about locations such

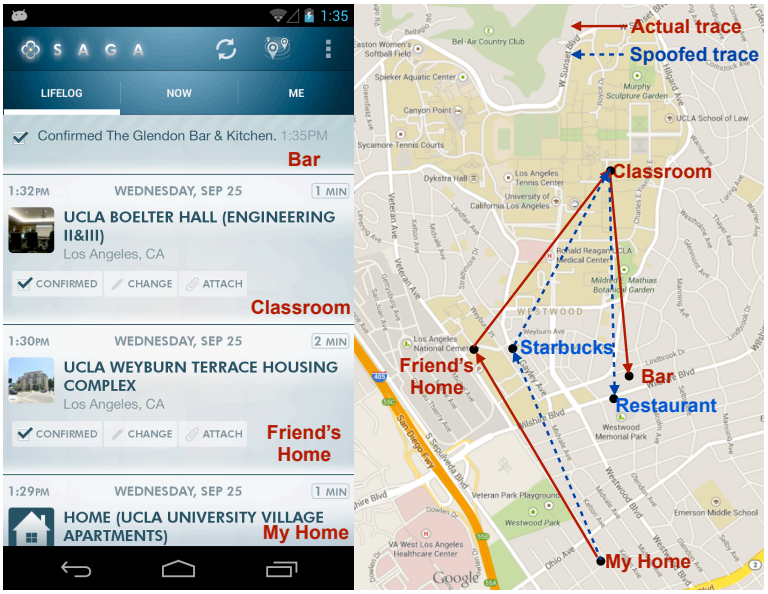


Figure 4.1: Left: Saga app showing actual trace of the user. Right: Both actual trace and spoofed trace on the map.

as *home, office, hospital* private to the user. Fig. 4.1(left) shows a mobility trace recorded using Saga. The user starts from home, picks up her friend and drives to school for class; later she also visits a nearby bar and wants to keep the visit private. In addition to this direct privacy requirement, there is also an indirect privacy concern. Saga reveals the home location of the user's friend. The location information can be coupled with other online resources to identify the home owner, and infer that the friend had gone to the bar too. Thus, privacy of both the user and her friend is compromised, even though the friend is not using Saga. We therefore want ipShield to allow spoofing of location traces to protect visits to sensitive places. A plausible spoofed trace is shown on the map in Fig. 4.1(right). We illustrate how ipShield achieves this in Section 4.8.

4.3 Revisiting The Inference Privacy Problem

Inferences are labels associated with data. We group labels of a similar (semantic) type into an *inference category*. The category names and the grouping are based on prior work (Table 4.3). For example, *hospital, home, office* are grouped under Location category. An adversary tries to infer/predict these labels from the shared data. The prediction accuracy of an inference category corresponds to correctly predicting a label in that category. We now define the inference privacy problem.

Problem statement: Data is typically shared with an app for a specific set of inference categories. For example, in Section 4.2.1, data is shared for inferring the Transportation Mode, and in Section 4.2.2 it is for inferring travel statistics. These categories and their labels form a whitelist which the user wants to allow. However, the same data can be used to infer keystrokes and sensitive locations – inferences sensitive to the user. The sensitive categories and their labels form the blacklist which the user wants to keep private. The privacy problem is to design a system which will take as input the whitelist and blacklist of inference categories and translate them into privacy actions on the shared sensors such that the conditions on the lists are satisfied.

Side-Channel Attacks: Traditionally side channel attacks are ones which are designed to exploit the information revealed by execution of a cryptographic algorithm to recover the secret key. Such information channels include but are not limited to running time, cache behavior, power consumption pattern, acoustic emanations and timing information [KB07, SWT01, GST13]. Sometimes, even without any algorithm execution, information side channels exist due to physical signals emanating from a hardware while being used by a user. For example, acoustic [AA04, MVC11] and electromagnetic [VP09] emanations from a keyboard has been used to infer keystrokes and recover sensitive passwords and PINs. The feasibility of such attacks on the smart phone using sensor data has been evaluated

Privacy Analysis	Kirin [EOM09], SOM [BKO10], Stowaway [FCH11]
Privacy Detection	**Static**: BlueSeal [HMN]
	Dynamic: TaintDroid [EGC10]
Privacy Mitigation	**Mobile Based**: Dr. Android Mr. Hide [JMV11], PMP [AH13], Apex [NKZ10], MockDroid [BRS11], AppFence [HHJ11], pDroid [pdr], πBox [LWG13]
	Cloud Based: Lockr [TSG09], PDV [MHM10], Persona [BBS09]

Table 4.1: Categorization of prior work.

in [ASB12, MVB12].

Our inference privacy problem differs from traditional side-channel attacks in several ways. First, the shared data used for the attack are not unintended physical signals emanated from the hardware, or covert timing information but sensor data intended for the recipient. Second, in our setting the recipient is also the adversary whereas in case of side-channel attacks the adversary is typically different from the intended recipient. Finally, at least in principle the side-channel attacks can be prevented by placing the computational hardware in physically isolated and secure chamber whose boundaries the electromagnetic, acoustic and such emanations cannot cross which is not the case in our scenario. In [ASB12], inferring the keystrokes is referred as a side-channel attack. However, we call it a blacklist inference as the sensor data are intended to be shared with the app for the whitelisted inferences and are not a side-channel.

4.4 Related Work

We group prior work on systems for protecting privacy under three broad categories as shown in Table 4.1. The Privacy Analysis category summarizes contri-

butions towards analysis of the Android permission model, conformance of the various apps to this model, the usage pattern of permissions across apps, and finally the expressibility of the permission model [EOM09, BKO10, FCH11]. Under Privacy Detection we have tools such as BlueSeal [HMN] which use static analysis of the app bytecode to detect if sensitive information is being leaked over the network interface and inform it to the user at install time. Other systems like TaintDroid [EGC10] use dynamic flow tracking to detect malicious app behavior. However, both techniques can only alert the user of malicious behavior (BlueSeal at install time, and TaintDroid at runtime), and do not provide suitable mechanisms to prevent information leakage.

Under the Privacy Mitigation category we have mobile implementations such as Dr. Android and Mr. Hide [JMV11] that use instrumentation of the dex bytecode to ensure that access to all private resources is made available only through their trusted interface. AppFence [HHJ11] builds on TaintDroid to provide *shadow* or synthetic data to untrusted apps and measure the effect of such data on app utility. Other systems in [BRS11, pdr, NKZ10, AH13, LWG13] provide users with the ability to control access to their resources at runtime - a feature that is currently being integrated into the latest Android release [and]. However, the above systems provide binary access control to resources, do not monitor access to innocuous sensors, and lack higher level user-understandable privacy abstractions. Cloud-based solutions in [TSG09, MHM10, BBS09] are for protecting privacy of data streams but require additional infrastructure. A detailed exposition of other privacy techniques, and initial ideas on ipShield can also be found in [CRJ13].

4.5 Background: Android

Below we describe data paths, from sensors to apps and also highlight Android's security model [sec] to understand the process level isolation of the components.

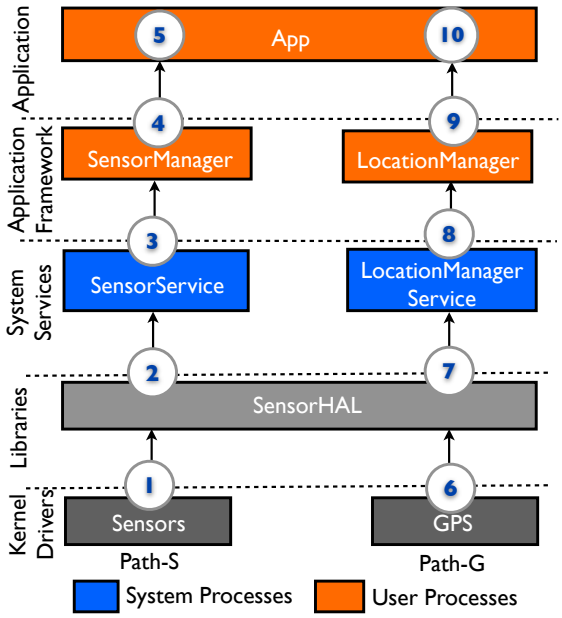


Figure 4.2: The data flow path from various sensors to apps.

4.5.1 Android Sensor Data Flow Path

We consider two data paths as shown in Fig. 4.2. Path-S is used by sensors such as accelerometer, gyroscope, light and so on. Path-G is from the GPS to apps. Note that the paths are simplified representation showing only the components of the Android OS that are relevant to ipShield. `SensorService` and `LocationManagerService` are system services (running continuously in the background) and are started by the Android OS at boot time. They run as separate threads within the same `system_server` process. These services poll the `SensorHAL` layer for sensor data and in turn pushes the data to the apps. The apps typically do not communicate directly with the services. Each system service has a corresponding `Manager` which acts as its proxy at the user level. Thus, each app instantiates either a `SensorManager` or a `LocationManager` object using the public API and then uses it to obtain the sensor data. As shown in Fig. 4.2, both the app and the manager objects are part of the same process.

4.5.2 Android Security Model

Application Sandboxing: The core of the Android OS is built on top of the Linux kernel, and this allows Android to re-purpose the traditional security controls built into Linux to protect user data, system resources, and to provide app isolation. Android enforces kernel level *Application Sandboxing* for every software that runs above the kernel which includes all apps, OS libraries, OS-provided app framework and app runtime. The Android system sets up the sandbox and enforces security between apps by assigning a unique user ID (UID) to each app and by running it as that user in a separate process. Running apps within a sandbox environment ensures that any memory corruption error will only allow arbitrary code execution in the context of that particular app and with the permissions established by the OS. User-specific privileges also ensure that files created by one app cannot be read or altered by another app.

Secure IPC: Android not only supports traditional mechanisms such as filesystem, sockets and signals but also implements newer and more secure mechanisms such as Binder and Intents.

Access Control Using Manifest: Finally, Android controls app access to resources by designating certain APIs (such as camera, location, bluetooth etc.) as protected [sec]. To use these resources an app needs to define its requirements in its manifest (a control file provided by every app). The user can either grant all of the requested permissions as a block or not install the app at all.

4.6 Architectural Design

The design of ipShield is guided by four objectives – better monitoring of sensor access, meaningful privacy abstraction, privacy rule recommendation and fine-grained control over shared data. The architectural requirements to achieve the

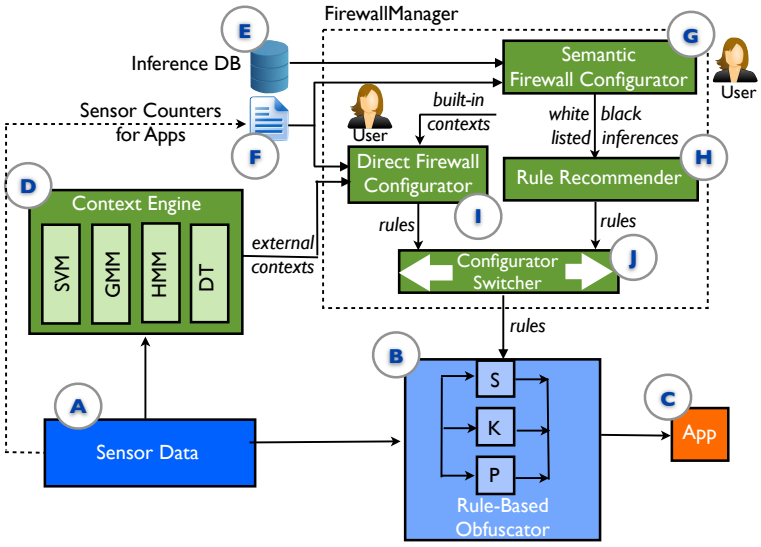


Figure 4.3: ipShield data flow.

above functionalities are shown in Fig. 4.3 and can be broken down into four major blocks – (i) Databases (ii) Context Engine (iii) FirewallManager (iv) Rule-Based Obfuscator. We describe each of the blocks and their components in detail below.

Databases: We maintain two databases: Sensor Counters and Inference DB. Currently, apps have unrestricted access to the class of innocuous sensors. One of our goals in ipShield is to instrument the OS to monitor the number of sensors accessed by an app. The information is populated in the Sensor Counters database (marked Ⓕ) and is provided as an input to the FirewallManager block. The database needs to be updated when a new sensor is accessed by an installed app or when an app in uninstalled.

Motivated by the database of virus signatures maintained by antivirus software, we maintain a similar database for mapping the list of inference categories (and their labels) that could be predicted using a combination of sensors, together with the prediction accuracy and the machine learning algorithm employed (Table 4.2 shows a small subset of the Inference DB). Advances in sensing coupled with increases in the sophistication of learning algorithms result in newer inference

categories and improved accuracy. The inference DB (marked Ⓔ) thus needs to be kept updated.

Context Engine: For granularity of rules, ipShield allows trusted Context Engines (marked Ⓓ) to register and provide as input context labels. A context engine is a set of machine learning algorithms, which take as input raw sensor data and output the current context label. An inference label is same as the context label but it is inferred from the shared data by the adversary. A user can configure privacy rules to trigger on context labels.

FirewallManager interacts with the user and is responsible for generating the privacy rules. There are four different sub-blocks within FirewallManager (marked Ⓖ through Ⓙ).

The Semantic Firewall Configurator (Ⓖ) takes as input the sensors accessed by an app and queries the Inference DB to present the user with a list of possible inference categories that can be predicted by the app. Using inferences instead of sensors allow us to better communicate the privacy risks to the user [RGK11]. The user then configures a whitelist and a disjoint blacklist from the enumerated list of inference categories.

The Rule Recommender (Ⓗ) (Section 4.6.2) takes as input the privacy preferences of the user expressed in terms of the whitelist and blacklist and translates them to actual privacy actions on the sensors. We observe that the privacy actions are dependent on the inference labels, the learning algorithm employed and the features used. Therefore, to keep the recommender simple and generic we limit the auto-generated privacy actions to Normal and Suppress (Section 4.6.1). While the auto-generated rules are binary and conservative, we provide the user with the flexibility to override them.

The Direct Firewall Configurator (Ⓘ) allows the user to manually configure fine-grained context-aware rules. The contexts used can either be ones provided by

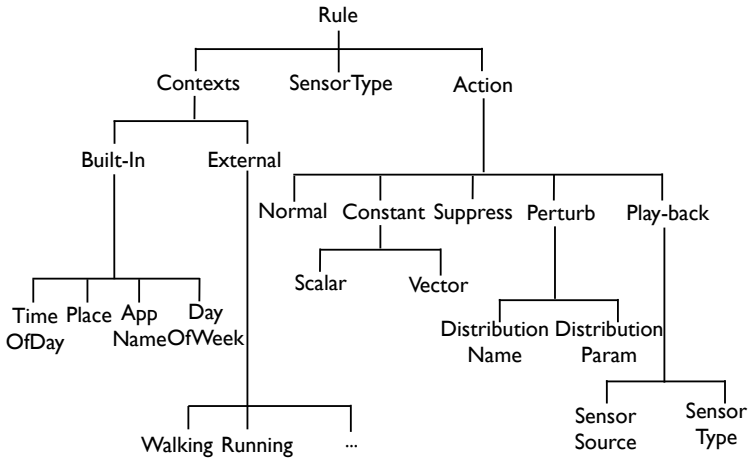


Figure 4.4: Tree showing all the possible options currently implemented in ipShield for constructing privacy rules. The leaf nodes of the tree are instantiated to form the privacy rules.

ipShield, or ones which are externally obtained from the trusted Context Engine. ipShield is designed to operate in the Semantic mode. The Direct Configurator is an optional mode, which provides flexibility of rule configuration at the cost of increased human interaction.

Finally, the Configurator Switcher block (Ⓙ) allows the user to switch between the Semantic and Direct Configurator modes and configure rules.

Rule-Based Obfuscator (Ⓑ) implements the different privacy actions. It takes as input privacy rules and sensor data and, depending on the app, applies the appropriate rules to the data before releasing them.

4.6.1 Taxonomy of Privacy Rules

The complete list of choices for configuring privacy rules is illustrated in Fig. 4.4. A rule has three basic parts: Context, SensorType, and Action. We also allow conjunction (denoted by the $\wedge$ operator) of the context labels within a rule. The general form of a rule is if $(\wedge_{i=1}^{n} Context_i)$ then apply $Action$ on $SensorType$. For

example, if $((TimeOfDay \text{ in } [10am - 5pm]) \wedge (Place = school) \wedge (AppName = facebook))$ then apply $Action = Suppress$ on $SensorType = gps$. As shown in Fig. 4.4 some of the simple contexts such as TimeOfDay, DayOfWeek, Place and AppName are built into ipShield. External contexts provided by a registered Context Engine can also be used to configure rules. SensorType refers to the sensor (e.g., accelerometer, GPS, gyroscope) on which the action is to be applied.

Excluding the default action of releasing data without any changes (Normal), ipShield currently supports four different privacy actions. The Suppress action (S-block in Ⓑ) when applied blocks data from reaching an app and the app is unable to detect any sensor event. The Constant action (K-block in Ⓑ) allows user to replace actual data with a constant value. The user-specified constant can be vector or scalar valued depending on the type of sensor whose data is being replaced. The Perturb action (P-block in Ⓑ) can be used to add noise to sensor data. The noise values can be drawn from different probability distributions, the parameters of which are input to this action. Finally, the Play-back action can be used to suppress the data from the actual sensor hardware and instead send synthetic sensor measurements from an external service to the requesting app (U-shaped datapath). The synthetic data source and sensor type are input to this action. The Play-back option can be used for generating any arbitrary transformation on the data offline.

4.6.2 Rule Recommender

The Rule Recommender takes as input the whitelist and the blacklist of inference categories and generates a configuration for enabling or blocking of sensors accessed by an app. The goal is to ensure that only those inference labels which form part of the whitelist are allowed and those in the blacklist are blocked.

Sensors	Inference Categories			Evaluation		
	Transport Mode	Location	Onscreen Taps	Priority1 $\{10, 4, 10\}$	Priority2 $\{10, 0, 7\}$	Priority3 $\{5, 9, 9\}$
GPS+ Acc + Gyro	95%	97%	80%	0	**869.4**	-875.8
GPS+WiFi	83.1%	97%	0%	**835.4**	849.9	-470.0
GPS+GSM	81.7%	98.2%	0%	820.9	835.6	-476.6
GSM+WiFi	72.9%	94.03%	0%	731.45	745.5	**-458.1**
GSM+Wifi +Acc+Gyro	92%	94.03%	80%	0	838.7	-861.6
Wifi+Acc+Gyro	91.1%	23.08%	80%	0	830.2	-498.6
GSM+Acc+Gyro	88.1%	94.03%	80%	0	798.8	-862.8
GPS	75.8%	97%	0%	760.7	775.2	-472.4
GSM	61.8%	94.03%	0%	617.8	631.9	-461.7
Acc+Gyro	84.6%	23.08%	80%	0	763.7	-500.7

Table 4.2: Left: A portion of the Inference DB (mapping M). Each entry (in %)is the maximum prediction accuracy for the inference category using the sensor combination. Right: The objective function (Eqn. 5.8) evaluated for different priority vectors and $P_{max} = 10$.

4.6.2.1 Problem Formulation

Let N be the number of sensors used by an app (obtained from Sensor Counters) and $\mathbf{s} = [s_1, \ldots, s_N]$ represent the sensor state vector where $s_i \in \{0, 1\}$ represents the state of the i^{th} sensor. Setting s_i to 0 indicates that the sensor is disabled and a value of 1 indicates that the sensor is enabled. We denote the set of inference categories by $\mathcal{L} = \{l_1, \ldots, l_{|\mathcal{L}|}\}$. We define a mapping $M : \{0, 1\}^N \times \mathcal{L} \rightarrow [0, 1]$ where $M(\psi, l) = 0$ indicates that there exists no learning algorithm which can use the data streams from the sensors which are enabled as per state vector ψ and infer a label in category l. A non-zero value of $M(\psi, l)$ correspond to the maximum accuracy among all the learning algorithms that can be used to infer category l from the data streams released as per the state vector ψ. A value of 1 indicates that l can be perfectly inferred using the enabled sensors. Note, we

might use different learning algorithms to predict the same labels using different sensor state vectors. The mapping M is obtained from the Inference DB. Learning algorithms typically work on features extracted from the raw sensor data. But, our current model is agnostic to features because we are sharing the raw sensor data itself and hence every required feature can be extracted from it. The set of whitelisted categories $\mathcal{W} \subseteq \mathcal{L}$, and the set of blacklisted categories $\mathcal{B} \subseteq \mathcal{L}$, are as specified by the users such that $\mathcal{W} \cap \mathcal{B} = \emptyset$. Finally, let $p_l \in \{0, \ldots, P_{max}\}$ denote the priority level set for category l by the user such that a higher value of p_l indicates higher priority. The priority levels represent a relative gradation of risk as perceived by the user. For example, $P_{max} = 3$ could correspond to *low*, *medium* and *high* levels of perceived risks. We use the above notations to formulate the inference-privacy problem as the following constrained optimization problem

$$\max_{\psi \in 2^N} \quad \sum_{l \in \mathcal{W}} M(\psi, l) 2^{p_l} - \sum_{l \in \mathcal{B}} M(\psi, l) 2^{p_l} \tag{4.1}$$

$$\text{s.t.} \qquad \sum_{\substack{l \in \mathcal{B} \\ p_l = P_{max}}} M(\psi, l) = 0. \tag{4.2}$$

The objective function in Eqn. 5.8 is designed to maximize the prediction accuracy of the whitelisted labels and minimize the prediction accuracy of the blacklisted labels. The priorities are exponentially scaled up to account for whitelisted labels which can be detected with low accuracy than other labels but have a higher priority. The constraint in Eqn. 5.9 ensures that users can force blacklisted inferences to be blocked by setting their priority to P_{max}. We note that the search space in the optimization problem shown in Eqn. 5.8 is constrained to the vector of elements with 0's and 1's corresponding to the enabled and blocked sensors respectively. It then follows that the search space is constrained to the vertices of a hypercube. It is also easy to show that this search space is non-convex. Moreover, the optimization function depends on the relation M on which we impose no structure or even linearity. Thus, our program is non-linear integer program which is non-convex and NP-complete. We observe from our investigation of prior

work and apps from Google Play (Section 4.8) that $N \leq 6$ for almost all the apps. Therefore, to solve a specific instance of the optimization problem above (a given choice of whitelist, blacklist, N, and priorities) we apply brute force and enumerate all possible state vector combinations. We filter out all state vectors which satisfy the blacklisted constraint and maximize the objective function over this reduced space. The output vector ψ shows which sensors should be enabled or disabled mapping preferences on inferences to privacy actions on sensors. There will be scalability issues for large N (> 15), but in practice, its improbable for a single inference to be using 15 different sensors on a phone in the near future.

4.6.2.2 Numerical Example

We return to the motivating example (Section 4.2.1) and express it in terms of the notation described above. Thus, $\mathcal{L} = \{\text{Transportation Mode}, \text{Location}, \text{Onscreen Taps}\}$, $\mathcal{W} = \{\text{Transportation Mode}\}$ and $\mathcal{B} = \{\text{Location}, \text{OnscreenTaps}\}$. The mapping M is presented in Table 4.2 (under Inference Categories). We set the maximum priority level $P_{max} = 10$ throughout this example and represent a user specified priority vector as a tuple $(p_{transport}, p_{location}, p_{tap})$. We apply the algorithm above for different choices of priority vectors and report the evaluation results also in Table 4.2 (under column titled Evaluation).

Consider $Priority1 = (10, 4, 10)$ as the selected priority vector. The user is not too concerned about revealing his Location and sets $p_{location}$ to 4. She however wants to strictly suppress the detection of Onscreen Taps and sets p_{tap} to 10. A high priority is also given to the whitelisted inference category by setting $p_{transport} = 10$. The objective function values for the different sensor combinations is shown in the column under heading Priority1. The maximum occurs for the combination corresponding to GPS+WiFi and is selected by the recommender. The selected sensor state vector is such that accelerometer data is suppressed in order to guarantee no leakage of the Onscreen Taps information. We also

note that GPS+WiFi configuration provides higher accuracy in predicting the Transportation Mode and lower accuracy for Location prediction compared to other sensor combinations.

We consider another scenario with priority vector $Priority2 = (10, 0, 7)$. In this case, the user does not worry about Location disclosure, but wants to increase the prediction accuracy of the Transportation Mode while blocking the Onscreen Taps if possible. The objective function values are shown under column Priority2. The recommender selects the GPS+Accelerometer combination which is biased towards performance. In addition to meeting blacklist requirements the combination also provides the best accuracy.

Finally, the third user has high levels of concern about revealing both Location and Onscreen Taps information. She would like to trade the performance with privacy and thus selects a priority vector $Priority3 = (5, 9, 9)$. The resulting sensor combination chosen is GSM+WiFi. The rule recommender starts by suppressing the accelerometer data (to prevent tap inference). It then selects the combination which results in the worst Location inference from among the remaining set of combinations (GSM and GSM+WiFi), while simultaneously maximizing the whitelist accuracy.

Model-Based Augmentation of Rule Recommender: Prior research has shown that a user's various context labels and transitions between them can be captured by a Markov chain [GNG12], by using a Dynamic Bayesian Network [PCG13], or explicitly enumerated [CBS13]. A user specifies a whitelist and a blacklist of inference categories, and depending on the current context label and the learned model the system can determine whether to release a context with a particular probability. In other words, the probability of release of a context should not increase the adversarial accuracy of predicting a blacklisted inference label. We envision that such model-based techniques can also be included in our recommendation system for generating a richer set of privacy rules.

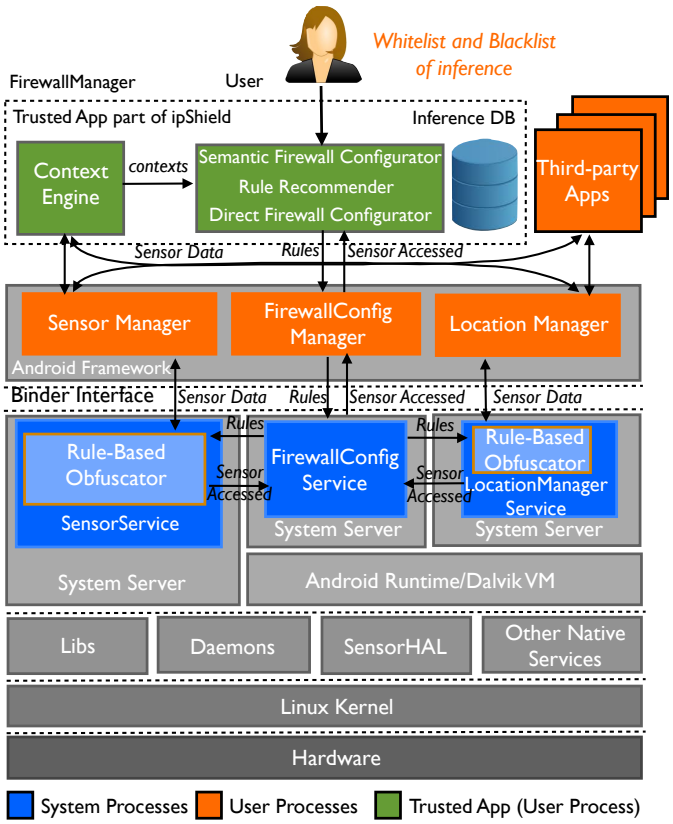


Figure 4.5: Implementation of ipShield on Android.

4.7 Implementation

The implementation of ipShield on the Android stack (Fig. 4.5) is described below.

4.7.1 Trust Model

We assume that the user installed third-party apps (e.g., from Google Play) are untrusted but do not collude with each other and share information. We trust the Linux kernel on which Android OS is built and also the Application Sandbox implemented by the kernel (Section 4.5). We extend the chain of trust to include the OS libraries and system services which run within the Application Sandbox and are protected by UID and group ID privileges. However, recent successful exploits from Facebook on modifying the internal data structures of the Dalvik

VM [buf] leads us to not trust the Application Framework components which run within the same process as the Dalvik VM.

4.7.2 Intercepting Data: Possible Choices

As indicated by the markers (①-⑩) in Fig. 4.2, there exist different operating points in both the data paths (Path-S and Path-G) at which we can intercept the sensor data, apply privacy actions, and obfuscate it. However, also associated with an operating point is the implementation complexity of the Rule-Based Obfuscator block (Ⓑ in Fig. 4.3) at that point and also its vulnerability to security attacks. We discuss below the trade offs in selecting an operating point.

Points ① and ⑥, correspond to modifying the kernel drivers to obfuscate data. While the drivers are protected by kernel security mechanisms, they require our implementation to be vendor specific. It is also hard to push app and rule information to the drivers and periodically update rules inside a driver.

Points ② and ⑦, correspond to changes in the SensorHAL layer. The HAL provides the abstraction between device specific kernel drivers and the Android system above. However, changing the HAL like the kernel driver has a high implementation complexity in terms of pushing app and rule information.

Points ③ and ⑧, correspond to modifying the Android system services, namely `SensorService` and `LocationManagerService` which are responsible for handling the different sensors and the GPS respectively. These services as shown in Fig. 4.2 run in a process separate from the app and hence are protected by the Application Sandbox. Both `SensorService` and `LocationManagerService` maintain information about installed apps, and can be easily signaled using binder calls and as we show later in Section 4.8 they incur low overhead while updating rules.

Similarly, points ④ and ⑨, correspond to changing the `SensorManager` and `LocationManager`, respectively. These points have the least implementation com-

plexity, however both `SensorManager` and `LocationManager` run within the same process as the app, and hence they are not protected by process-level isolation. Recent exploits have used the above vulnerability to modify code data structures at runtime [buf].

Finally, points ⑤ and ⑩, correspond to static analysis of the app code to understand privacy violations [HMN]. However, the information flow approaches are often conservative, incur large instrumentation and runtime overhead, and typically stop at identification of a malicious app. Based on the available choices we decided to implement the Rule-Based Obfuscator block in the `SensorService` and `LocationManagerService` blocks in the respective data paths.

4.7.3 ipShield Code Blocks

ipShield is an open source project. The code for each of the blocks together with complete instructions for downloading and installing ipShield are available at [ipS].

4.7.3.1 Databases

Sensor Counters: This database, implemented as a file, maintains a counter for each sensor on a per-app basis. The counter for a sensor represents the number of events from the sensor that have been sent to the app. We use an `unsigned` $64-bit$ `long int` for our counter. Even at the maximum sampling rate of a sensor, under continuous sensing, the counter will not overflow within the lifetime of a phone. The entry for an app together with the counters are deleted when the app is uninstalled from the phone. A counter value of zero indicates that the sensor is not being used by the app. These counters are maintained by {`Sensor, LocationManager`}Service and are periodically written to the `/data/sensor-counter` file every minute. The permissions on the file are such that it can be read by any app but can be written to only by system services.

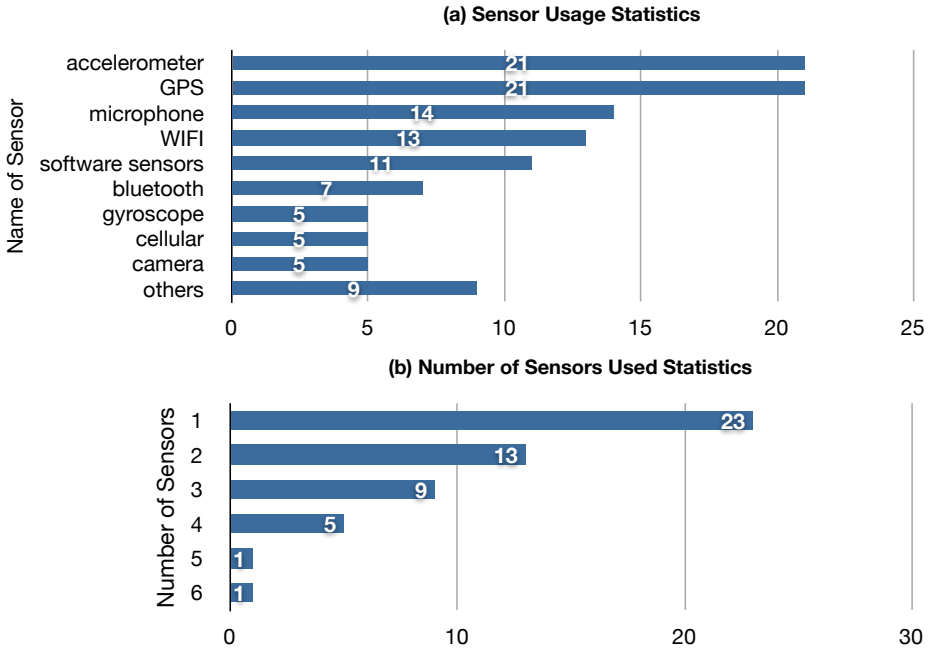


Figure 4.6: Statistics of sensor usage from the Inference DB.

Inference DB: A knowledge repository generated from a survey of 60+ papers published in relevant conferences and journals over the past $3-5$ years. This database captures a wide variety of inference categories a small set of which is shown in Table 4.3. For each inference category, we store the prediction accuracy over the constituents labels for a particular sensor combination. If there are multiple papers using the same sensor combinations predicting the same set of labels we store details of the one with highest accuracy. We also maintain information about the set of sensors used, the features extracted from the sensor data, the classifiers used, and finally the paper title under which the results were published. In Fig. 4.6, we show statistics of sensor usage computed using the inference database. Based on our survey, we found that (a) GPS and accelerometer sensors are the most commonly used; (b) the number of sensors accessed by any app is almost always less than 6 (we do not include papers which use external body worn sensors in the plot, but even externally worn sensors are less than 6 types). While newer inferences are being made, we do not expect the database

Inference Category	**Labels**
Transportation Mode [RMB10]	still, walking, motorized
Device Placement [PPC12]	hand, ear, pocket, bag
Onscreen Taps [MVB12]	location of taps on screen
Location [BW11] [KKE10] [NDA13]	home, work, public, restaurant...
Emotion [RMM10] [CC11]	happy, sad, fear, anger, neutral
Speaker [LJS12] [NDA13]	male/female, identity
Text Entered on Phone [MVB12]	alphabets
Stress [LFR12] [CC11]	stressful or not

Table 4.3: Selected inference categories from Inference DB.

to change rapidly. We envision crowdsourcing as a way to maintain an updated database. To enable that, we provide a web interface where people can contribute entries [ipS]. Currently, we rely on manual screening of the received entries before adding them to the Inference DB.

4.7.3.2 Context Engine

To allow fine-grained context-aware rules, ipShield allows trusted external context engines to register contexts that they can provide using the interface in Fig. 5.2(e). The user can then configure rules which will be triggered on a particular context. ipShield expects the context engines to use Android supported intents (`action`=label) as the IPC mechanism for providing the context labels.

Contexts such as battery status, contact list, ringer status etc., do not require access to sensor data and can be obtained through APIs provided by the Android OS. However, for contexts that require sensor data, the external context engine must have access to raw sensor data. To implement this when a data buffer from the HAL is received by the `SensorService` and/or the `LocationManagerService` it is first sent to the context engine to get the current context label. On receiving the context, the associated rules are then loaded and used by the Rule-Based

Obfuscator to obfuscate the data buffer.

We modified the Transportation Mode app [RMB10] to implement an activity context engine and test its integration with ipShield. In our implementation, the context engine used `SensorManager` for subscribing to accelerometer data at the rate of *SENSOR_DELAY_GAME*. This resulted in sensor data at a rate of $50Hz$ or a sample every $0.02s$. We used data buffered over a sliding window of $1s$ for inferring the activity context. On an average, the engine took about $8ms$ to generate activity context from a $1s$ accelerometer window. Even with additional overhead due to binder call and rule loading, we found that the associated rules can take effect before the next sensor data sample. This meant that our buffer size could be equal to $1s$ of data without losing any sample. In general, for keeping the buffer size bounded we observe that the processing time of the context engine together with the rule update time should be less than the inter arrival time between two data samples.

4.7.3.3 FirewallManager

The FirewallManager is a trusted Android app which has three different components described below.

Semantic Firewall Configurator: This is an Android activity. It reads the Sensor Counters for the installed apps and queries the inference DB for possible inference categories for each app. When launched it displays this information (Fig. 5.2 (a)) for the user. Once the user selects an app she is presented with the inference categories with an option to classify each into a whitelist or a blacklist (Fig. 5.2 (b)). The Configurator then passes the data user preferences to the Rule Recommender.

Rule Recommender: The algorithmic aspects of the rule recommender are described in detail in Section 4.6.2. It is implemented within the Seman-

tic Firewall Configurator. It then uses the `FirewallConfigManager` to write the rules to `/data/firewall-config` file and also use a binder call to signal the `SensorService` and `LocationManagerService` to reload the new rules.

Direct Firewall Configurator: In this mode the user can configure context-aware privacy rules (Fig. 5.2 (c) and (d)). The user can specify actions on sensors used by apps, and for each action also associate either built-in contexts such as TimeOfDay, DayOfWeek, Place, or external contexts as triggers. For defining the Place context, the user can drop a marker on the map as shown in Fig. 5.2 (f) to annotate a $< latitude, longitude >$ tuple with a significant place name. For external contexts the Configurator implements a `BroadcastReceiver` which listens for intents. When an intent containing a particular label is received, the `BroadcastReceiver` invokes a rule loader service which passes a pre-configured set of rules associated with the label to the `FirewallConfigManager`. The `FirewallConfigManager` writes the rules into a file `/data/firewall-config` and signals both `SensorService` and `LocationManagerService` to reload the new rules. Note that the user can also explicitly request for loading a new set of rules.

4.7.3.4 Rule-Based Obfuscator

The Rule-Based Obfuscator block is responsible for enforcing the actions specified by the privacy rules described in Section 4.6.1. This block is implemented both within `LocationManagerService` and `SensorService` with the same functionality. The rules are read from the `/data/firewall-config` file and inserted into a HashMap for faster access. The serialization and deserialization of both rules and Sensor Counter is implemented using Google Protocol Buffer [pro]. The hash for each rule is computed on the fields {`appName, UID, sensorType, ruleSeqNum`} where `ruleSeqNum` is a sequence number assigned to a rule for a sensorType. This allows multiple rules for a sensor implementing the OR operation on contexts (AND operation is implemented by allowing multiple contexts for each rule).

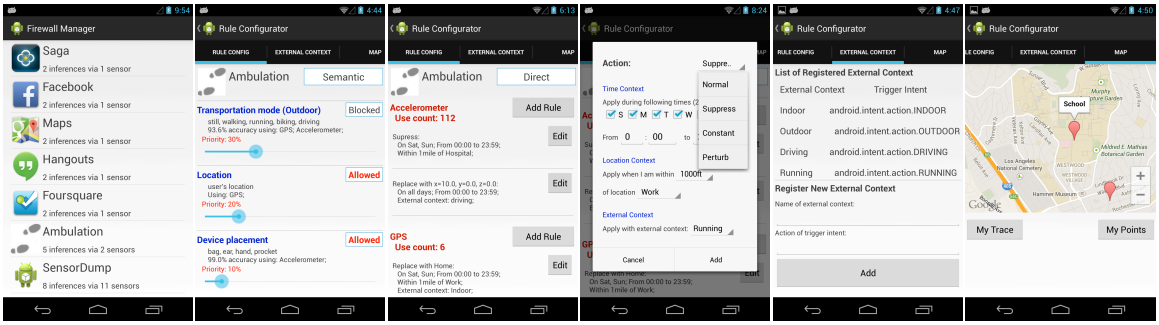


Figure 4.7: (a) List of installed apps showing number of sensors and number of possible inferences. (b) Semantic Firewall Configurator showing list of inference categories with option to block or allow. (c) List of rules configured for different sensors. Multiple rules with combination of contexts can be configured for each sensor. (d) Direct Firewall Configurator for privacy actions and their parameters. (e) List of external contexts registered with FirewallManager and ability to add new ones. (f) Screen to annotate significant places on the map (provides built-in Location context for rules).

UID is assigned to an app by Android at install time.

The `FirewallConfigManager` interfaces with both the Semantic and the Direct Configurator modules of FirewallManager app and communicates the privacy rules to the `FirewallConfigService` through the binder interface. The service runs within a system process and writes the rules to the `/data/firewall-config` file and signals the `LocationManagerService` as well as the `SensorService` to reload the new rules.

4.8 Evaluation

We implemented ipShield by modifying the Android Open Source Project [aos] (AOSP, branch 4.2.2_r1). We deployed and performed all our tests on the Google Nexus 4 phone (1.5GHz quad-core Qualcomm Snapdragon™ Pro, 2GB RAM).

4.8.1 Performance Overhead

We measured the overhead incurred by running ipShield to highlight that it is feasible to deploy it on current mobile platforms without impacting user experience in terms of battery life and app responsiveness.

4.8.1.1 Rule Access

Android supports four different sampling rates. On the Nexus 4 we found that on average SENSOR_DELAY_NORMAL and SENSOR_DELAY_UI are less than 10Hz, SENSOR_DELAY_GAME is around 50Hz and SENSOR_DELAY_FASTEST is around 200Hz. In Fig. 4.8 (a), the blue bars show the times taken to load the rules from the file (`/data/firewall-config`) into the HashMap, which are negligible. The green bars in the figure represent total time for the rules to take effect after configuration. For SENSOR_DELAY_NORMAL and SENSOR_DELAY_UI no data sample will be released before the new rules take effect even for 200 rules. In reality, we believe that the number of privacy rules will typically be less than 50, therefore for SENSOR_DELAY_GAME and SENSOR_DELAY_FASTEST less than 2 and 6 samples will be released before the 50 rules take effect, respectively.

4.8.1.2 Sensor Data Access

The overhead i.e., difference in time for fetching one data sample using ipShield compared to that on unmodified AOSP is shown in Fig. 4.8 (b). The overhead is computed by taking the average of fetching 30000 samples. Each sensor is sampled at SENSOR_DELAY_FASTEST (200Hz). The time for ipShield is averaged over the time for performing each of Constant, Perturb, and Normal (no change) actions on every accessed sample. We can see that the access overhead per sample is less than $20\mu sec$ – negligible even for the fastest sampling rate.

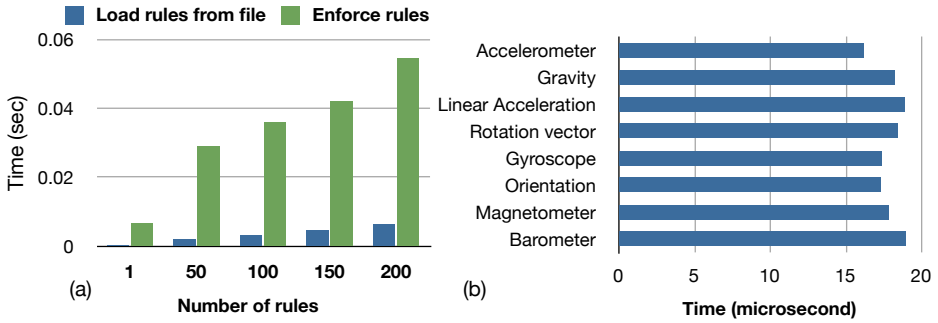


Figure 4.8: (a) Time taken in for the rules to load into memory and take effect. (b) Time overhead to fetch one sensor data sample sampled at SENSOR_DELAY_FASTEST.

4.8.1.3 CPU and Memory Overhead

The Rule-Based Obfuscator block is included inside both the `SensorService` and the `LocationManagerService` components that run as threads inside the `system_server` process. For each data sample, the Rule-Based Obfuscator block is called to apply the privacy actions. We compare the overhead of the Obfuscator block with AOSP by profiling the average CPU utilization of the phone while running the Ambulation app [RMB10] which continuously requests sensor data (GPS, accelerometer) at a rate of 1Hz on a Nexus 4 phone. CPU utilization with AOSP averaged 2%. CPU utilization with various privacy actions averaged 2.5% and never exceeded 3%. It should be noted that the CPU utilization (and hence energy consumption) will scale with the sampling rate. As shown in the energy analysis that follows this section, we believe that the overhead of ipShield is small enough to have negligible effects on overall system performance.

Memory overhead for the transformations is shown in Fig. 4.9 (a). The highest overhead is for Perturb and is less than 0.5MB. There is a dip in memory usage for Suppress action which lowers the average memory overhead over all the operations in ipShield to around 0.07MB.

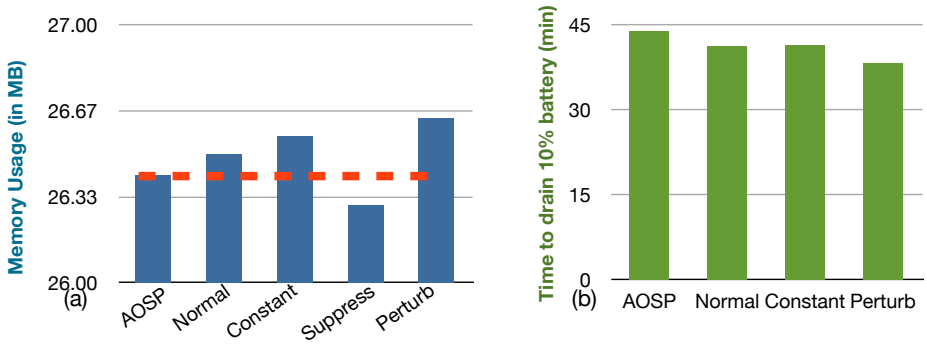


Figure 4.9: (a) Memory Overhead. (b) Energy Overhead.

4.8.1.4 Energy Overhead

We compare the energy overhead of ipShield to AOSP by plotting the time to drain the phone battery from 100% to 90%, while the Ambulation app is continually running in the foreground for Transportation Mode inference. All network interfaces and radios are turned off, and the screen display is on at the lowest brightness. We acquire a CPU wakelock in the app to prevent the phone from sleeping. The inference frequency of the app is set to 4Hz. We measure the drain due to three actions: Normal, Constant, and Perturb which will consume more power than the AOSP. Fig. 4.9 (b) shows the results: ipShield on average drains the battery 3min 37s ($\sim$ 8.2%) faster than AOSP, which we consider as a marginal overhead. In typical usage scenarios where the screen is at a higher brightness setting and the network subsystem is active we expect the energy overhead for ipShield to be relatively lower.

4.8.2 Vulnerability of Current Apps

We did a survey of the top 60 free apps from Google play store to find the different sensors used by these apps. We installed and executed each of the apps from the play store, and noted the permissions to sensors requested by the apps at install time, and also the sensors which were being accessed without permission using

Sensors	App Name	Inference Categories
GPS	Twitter, Amazon, eBay, Google Earth (and 15 more..)	A1: Loc, Speed, Route
Acc	Despicable me, Subway Surfers, Accupedo Pedometer	A2: TM, Device Placement, Text on Keyboard
Audio	Snapchat, Vine, Cadiograph	A3: Speaker, Loc, Emotions, Stress
Acc + GPS	Picsart, Temple Run	A1 + A2
GPS + Audio	FB, Tango, Whatsapp, Shazam, GoSMS	A1 + A3
Acc + GPS + Audio	Instagram, Neon motocross	A1 + A2 + A3
Acc + Pro + GPS + Audio	Skype	A1 + A2 + A3
Acc + Rot + GPS	Maps	A1 + A2, Onscreen Taps, Text Entered on Phone
Acc + Gyro + Pre + GPS	Saga Lifelogging	A1 + A2, Onscreen Taps, Text Entered on Phone

Table 4.4: Sensors and possible inferences from top apps in Google Play Store (all have access to network). Loc:Location, Acc:acclerometer, Pro:proximity, Rot:rotation vector, Gyro:gyroscope, Pre:pressure, TM:Transportation Mode.

ipShield. We also made use of the description of the app provided at the app store for additional information (if any). This provided the list of sensors used by each app. We then used the Inference DB to create the association between app and possible inference categories as shown in Table 4.4. The results from this survey validates our claim that GPS and accelerometer which are the most used sensors in academic research (Fig. 4.6) are also the most widely used sensors in apps. We further note, that many of these apps have access to data from innocuous sensors, combinations of which can be maliciously used to predict a lot more inferences than what they advertise.

4.8.3 Case Studies: Revisited

We now illustrate how ipShield can be used to configure simple rules to overcome the privacy issues outlined in the examples in Section 5.2.

***Transportation Mode and KeyLogging*:** While suppressing accelerometer at all times is a naive solution, to obtain better utility from the app, the user can use the Direct Rule Configurator to select an external context $KEYBOARD\_UP$, and use it to define the following rules: If $((TimeOfDay$ in $[12am - 11:59pm]) \wedge (ExternalContext = KEYBOARD\_UP) \wedge (AppName = Ambulation))$ then apply $action = Suppress$ on $SensorType = acc$; and a similar rule for suppressing $SensorType = gyro$. We exploit the fact that it is sufficient to block the accelerometer and gyroscope data while the softkeyboard is active to protect against keylogging. On executing the above rules, the act of suppression will inform an adversary that the user is entering text, but she cannot infer anything more. The Ambulation app will now continue to work at all times when the user is not entering text, maximizing its utility to the user.

***Saga and Location*:** A user would often like to keep some of his visits to sensitive places private. ipShield allows the user to configure the following rules to spoof her location trace: (1) If $((TimeOfDay$ in $[12am - 11:59pm]) \wedge (Place = Friend'sHome) \wedge (AppName = Saga))$ then apply $action = Constant$ and $value = Starbucks$ on $SensorType = GPS$; (2) If $((TimeOfDay$ in $[12am - 11:59pm]) \wedge (Place = Bar) \wedge (AppName = Saga))$ then apply $action = Constant$ and $value = Restaurant$ on $SensorType = GPS$; As we mentioned earlier user can configure labels such as *Starbucks, friend's home, bar* using the map interface in ipShield. To ensure plausibility of the shared location data the perturbation performed, or even the constant value provided, should conform to a map [Kru09].

4.9 Discussion

While phones have evolved into sophisticated sensing platforms the corresponding sensing stack where starting at the raw sensor data meaningful data abstractions are created at each layer (akin to a communication stack) [ES10] has not yet taken shape. Efforts like CondOS [CKL11], together with architectural changes such as dedicated co-processors for context detection [app] are steps in the direction towards introducing greater semantic clarity for shared data. With such a stack in place it is then a natural design choice to have a privacy system built within the OS itself exploiting the semantic granularity of data for improved privacy.

With ipShield we advocated the above design philosophy and took the first step towards creating a framework with architectural changes built within the Android OS to protect user privacy. We introduced better monitoring of accessed resources, proposed a user-understandable privacy abstraction in the form of possible inferences, allowed users to configure semantic privacy rules, and ensured that user preferences are securely enforced.

Orthogonal to the enforcement of rules is their creation. In future, to minimize user interaction in rule formulation it is imperative that systems are able to learn rules based on the semantic similarity of shared data and basic user preferences. With respect to granularity of rules, even with user participation privacy rules can often tend to become conservative impacting the app utility. To this end, careful integration of ipShield with various static analysis tools [HMN] could provide better insight into the working of apps and in the creation of balanced rules.

The other pertinent question is regarding the selection of a suitable set of privacy actions. Integration of cryptographic solutions would enrich the spectrum of available actions. In addition, currently ipShield does not handle traditional side-channels attacks and it will be an interesting extension to the current system.

Finally, any such system should be able to run on resource constrained plat-

forms. Our experiments with ipShield indicate that it has low performance overhead and can run continuously on various mobile platforms without impacting app responsiveness.

CHAPTER 5

iDeceit: A Privacy Framework For Model-Based Falsification of Location Traces

5.1 Introduction

Location-aware third-party apps often rely on continuous collection of location data for providing useful services. The data collected is used by the apps to improve their loading times [YCG12], provide tailored services [GNG12, tas, onx] and for efficient life-logging [sag, app]. However, some of the visited locations can be sensitive to the user and need to be kept private. This leads to a tradeoff where the user wants to share data to maximize the *utility* offered by the location-aware apps, but also desires the *privacy* of the sensitive locations to be maintained.

Several approaches have been proposed to address the above privacy problem. One is to break the association between the sensitive data and the user. This can be done by anonymizing the meta-data (e.g., phone identifiers such as IMEI and MEID numbers, personal identifiers) associated with the shared location data [ZZJ11, Swe02, MKG07]. However, as recent research has demonstrated that using *sufficient* auxiliary information it is often possible to deanonymize data [SH12, NS08, Ohm09, GP09, NS10]. Moreover, in many domains (e.g., medical, insurance), user identity is an integral part of the shared data and anonymization may lead to significant utility loss.

Another approach is to add random noise to the data samples [STT12, STL11, Kru09]. This is inadequate because successive location samples are often corre-

lated in both time and space [PCG13, GNG12]. An adversary can track the temporal transitions between the locations, detect inconsistencies in the noisy data and exploit them to perform denoising. Adding noise also reduces the utility of shared data.

A third approach proposes to selectively suppress data whenever the user transitions to a sensitive location. However, this naive strategy does not work as the very act of suppression reveals information. A modified scheme, adopted in [GNG12], is to probabilistically suppress data, i.e., always suppress when in a sensitive location and with a non-zero probability suppress when in a non-sensitive location. However, probabilistic suppression can often be conservative and result in loss of app utility especially when the user is not in a sensitive location (see Section 5.9). In addition, suppression of data can raise suspicion of possible obfuscation being performed by the user prompting a change in adversarial strategy.

To overcome the inadequacies in the above approaches a desirable strategy is to substitute the sensitive location segment with another that is safe (i.e., not sensitive). However, there are several challenges to implementing such a strategy. First, the selected safe segment should be plausible or consistent with the previously released data to avoid identification. Second, the falsified trajectory should be able to track and if possible converge to the actual user trajectory. This is because every safe location that is falsified for reasons of plausibility leads to utility loss. Finally, suitable metrics are required to systematically select one from among multiple candidate safe segments for substitution.

We present iDeceit, a model-based falsification framework that implements substitution of data segments with synthetic data to maintain privacy of sensitive locations while ensuring the plausibility of the entire location trace. To maintain plausibility of substituted data, iDeceit employs a user-behavior model that captures the temporal correlation between the locations visited by the user over time. Whenever a user visits a sensitive location the model is used to identify safe

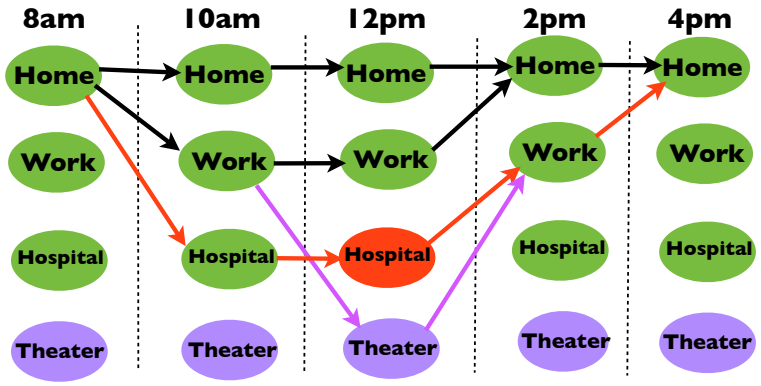


Figure 5.1: Example of a model using locations visited by a user over time.

candidate locations for substitution. By defining suitable utility, privacy and a novel plausibility metric, an optimal candidate location is chosen for substitution. Every subsequent release of data ensures that the entire falsified trajectory is still plausible and follows the actual user trajectory for possible convergence over time. To the best of our knowledge, ours is the first work to demonstrate the feasibility of model-based substitution for plausible falsification of location traces.

Plausible substitution offers another significant advantage. If done carefully, it can ensure minimal or no loss to privacy. In fact, we formally prove that using our model-based substitution ensures zero loss to privacy. This compares favorably with δ-privacy that has been used traditionally in location privacy research [GNG12]. We implemented iDeceit on the Android OS by introducing a new data flow path for pushing synthetic sensor data to the apps. Applying iDeceipt to substitute randomly chosen sensitive locations in real-life location data collected from a week-long study with 22 participants shows that iDeceit releases data for paths that are plausible (around 80%) as per user behavior, while providing maximum utility (around 90%), under zero-loss privacy.

5.2 Case Study

A user shares her location data with an app for accurate life-logging. Figure 5.1 illustrates the temporal behavior of the user over time. In particular, the nodes in green marked {*Home, Work, Hospital*} are the possible locations visited by the user over time. An edge between two nodes imply a possible transition, and although not shown in the figure, each edge is annotated with a probability value. Consider the trajectory marked by red edges in Figure 5.1. This trajectory consists of $path_0$ ={*(8am, Home), (10am, Hospital), (12pm, Hospital), (2pm, Work), (4pm, Home)*}. Now assume that the user has privacy concerns about the app provider knowing that she visits the Hospital and stays there for long time. Accordingly, she marks the state (12pm, Hospital) as a sensitive state. The main objective of iDeceit is to release (and artificially synthesize if needed) data that hide the fact that the user is present at the Hospital at 12pm while simultaneously ensuring that maximum number of locations are accurately reported to the app.

We make multiple observations. First, although {*Hospital at 10am*} is a safe state we cannot release it because typically if the user visits the *Hospital* at $10am$ she always stays there till $12pm$ – revealing the sensitive state. Thus, no released path should include the {*Hospital at 10am*} node. We use this intuition to remove all nodes and edges that can lead to any sensitive node. In the resulting residual graph every path is safe ensuring zero-loss privacy.

Second, among the green nodes there are two choices for selecting falsified but plausible paths: $path_1$ ={*(8am, Home), (10am, Home), (12pm, Home), (2pm, Home), (4pm, Home)*} or $path_2$ ={*(8am, Home), (10am, Work), (12pm, Work), (2pm, Home), (4pm, Home)*}. iDeceit needs to decide which path is better to release. Thus, we need a metric to select between competing paths while accounting for the typical user behavior. For example, if the user typically goes to *Work* between $10am - 12pm$ then *path2* should be chosen over *path1*. We define a novel

plausibility metric to quantify this notion.

Finally, none of the nodes in $path_1$ or $path_2$ match the nodes visited by the user in $path_0$ reducing the utility of the falsified path. To improve utility and increase choices available paths we merge together multiple user graphs into a larger super graph and search paths in this super graph. In Figure 5.1, the nodes marked $\{Theater\}$ together with the edges in purple correspond to information obtained from another user graph. In the super graph, $path_4$ ={*(8am,Home), (10am, Work), (12pm, Theater), (2pm, Work), (4pm, Home)*} matches the visited node *Work* at time $2pm$. Thus, $path_4$ provides greater utility than paths $path_1$ and $path_2$ but has lower plausibility as it includes nodes not present in the original user graph. This tradeoff between plausibility and utility in competing paths is the optimization problem we solve.

5.3 Solution Approach

An overview of the components required to implement iDeceit is described below.

5.3.1 Trusted Server

The server is used to upload user data and perform offline training of models that capture the temporal transitions between locations. It has an aggregate view of the data from different users which is used to train a *super* model containing all the locations and transitions for all users. While generating the *super* model the server also creates a dictionary that establishes a unique name space for the different locations across all users.

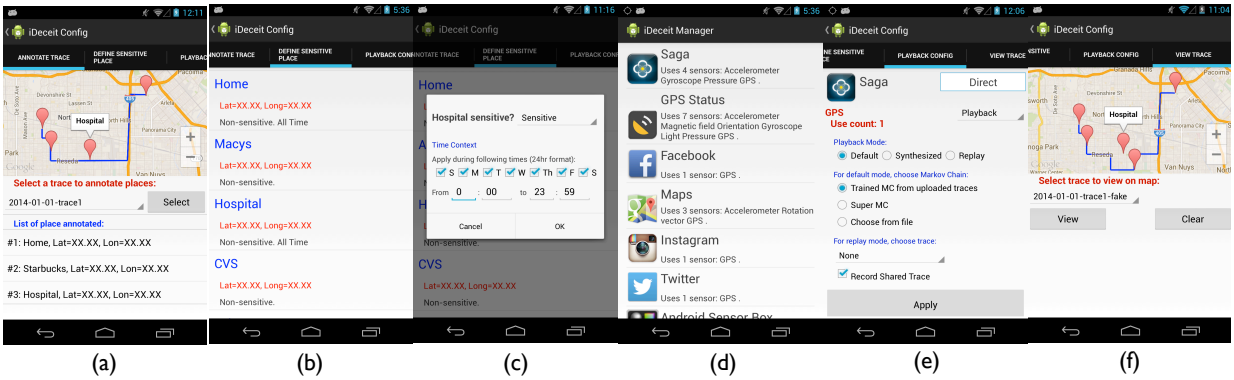


Figure 5.2: (a) The map interface is used to annotate locations (b)-(c) Time dependent sensitive locations are selected from the list of pre-configured locations. (d) List of installed apps and the sensors they access is shown. (e) Rule configuration page allows the user to configure the playback option for sensors at an app level. (f) Finally, falsified traces can be recorded and viewed by the user.

5.3.2 Privacy-Aware Model Generator (PAG)

The *super* model is a population-scale model trained using data from all users. The PAG block customizes the super model for a specific user. In addition, each user specifies some nodes in the model to be sensitive. PAG utilize this knowledge about the sensitive locations and generates a safe *residual* model by removing all the nodes that are sensitive or can lead to a sensitive node.

5.3.3 Path Planner (PP)

The PP block determines whether or not iDeceit needs to falsify (fabricate) the current location. It implements algorithms that take as input the current location, the previous released location, the *residual* (safe) model and the user model. The output of the algorithm is the next safe state that can be released.

5.3.4 User Interaction

To motivate the system architecture we begin by describing the end-to-end user interaction with iDeceit (see Figure 5.2). The user starts by annotating locations for a previously saved trace (Figure 5.2(a)). These annotated traces can then be uploaded to the server for constructing the model. Next, the user can specify time dependent sensitive locations from among the configured ones (Figure 5.2(b) and (c)). Finally, from a list of currently installed apps on the phone, depending on the sensors they access (Figure 5.2(d)) the user can select apps and configure a playback option on their accessed sensors. The playback mechanism is responsible for pushing synthetic data generated for falsified paths to the apps. In addition to synthetic data the playback mechanism also allows the replay of data from a previously uploaded falsified location trace (Figure 5.2(e)). Falsified traces generated by iDeceit can also be recorded and viewed at a later time (Figure 5.2(f)).

5.4 Mathematical Definitions

In this section, we introduce our formal notation and define the various models and metrics.

5.4.1 User Model

A user transitions between different location over the course of a day. Our target is to build a mathematical model that is able to capture the temporal dependence between the transitions. In our framework, we utilize a first-order Markov chain to form the user model [GNG12].

Let $C^u = \{c_1^u, c_2^u, \dots c_{m_u}^u\}$ be the set of distinct locations for the user u and $\mathbb{T} = \{1, \dots, T\} \in \mathbb{N}$ represent the time instants over the day. Since we aim to model temporal correlation between transitions, we define the state space of the

Markov chain Ω^u as the Cartesian product between the locations C^u and the time T, i.e.

$$\Omega^u = C^u \times \mathbb{T} \cup \{start,\ end\},$$

where the *start* and *end* nodes are special states such that the user always starts and ends the day in those two states respectively.

It follows from the above definition of the state space that user context states are time dependent. For example, the context state "being at home at 9am" is different than the state "being at home at 9pm". Incorporating time as a component of the state allows us to convert the Markov chain into a directed acyclic graph (DAG) with $T+2$ stages and $|\mathbb{T}| \times |C^u| + 2$ different nodes (notation $|S|$ denotes the cardinality of the set S).

To differentiate between the locations at time t and those at time t', we use the notation Ω^u_τ to denote the restriction of the set Ω^u to the time $t = \tau$, i.e.,

$$\Omega^u_\tau = \{\omega^u_{i,t} = (c^u_i, t) \in \Omega^u | t = \tau, c^u_i \in C^u\}.$$

We adopt the notation $\omega^u_{i,t} \in \Omega^u_t$ to denote the state comprising of location i of the user at time t, i.e., $\omega^u_{i,t} = (c^u_i, t)$.

To model the transitions between the locations, we use:

1. The random variable X^u_t to represent the actual location of the user u at time t and x^u_t be the value taken by X^u_t, i.e.,

$$x^u_t \in \{\omega^u_{1,t}, \omega^u_{2,t}, \ldots, \omega^u_{m_u,t}\}.$$

2. The transition probability P^u_t between location $\omega^u_{i,t-1}$ and $\omega^u_{j,t}$ at time t is defined as:

$$P^u_t(\omega^u_{i,t-1}, \omega^u_{j,t}) = \mathbb{P}(X^u_t = \omega^u_{j,t} \mid X^u_{t-1} = \omega^u_{i,t-1}).$$

We define P^u to be the family of all transitions probabilities over all times. Hence, the user Markov chain can be defined as a pair $\mathbb{M}^u = (\Omega^u, P^u)$.

5.4.2 Augmented Sensitive Locations

The user defines some locations to be sensitive (private). We denote by $\Omega_p^u \subset \Omega^u$ the set of sensitive locations that the user wants to keep private.

We augment Ω_p^u with additional locations. These locations, when visited, leaks significant information about the sensitive locations. We denote this new set by Ω_{p+}^u and we define it as:

$$\Omega_{p+}^u = \Omega_p^u \cup \left\{ x_{t-k}^u \mid \prod_{j=k}^{1} P_{t-j+1}^u(x_{t-j}^u, x_{t-j+1}^u) = 1, 1 \leq k < t \right\}$$

Following the notation of Ω_t^u, we define $\Omega_{t,p+}^u$ as the restriction of Ω_{p+}^u to time t (i.e., sensitive locations at time t).

5.4.3 Released Information Model

iDeceit monitors the current location of the user x_t^u. It then decides whether to release or synthesize (fabricate) location data before sharing it with the untrusted apps. Let random variable $\widetilde{X}_t^u \in \Omega_r^u$, be the released location by iDeceit at time t and $\tilde{x}_t^u$ the value taken by $\widetilde{X}_t^u$. Our objective is to ensure that the released random variable $\tilde{x}_t^u$ leaks zero information about the sensitive locations.

In order to achieve zero information leakage, the released locations is selected from the *residual* (or *allowed*) Markov chain, denoted by $\mathbb{M}_r^u = (\Omega_r^u, P_r^u)$. As the name implies, the *residual* Markov chain is constructed by removing the set of extended sensitive locations Ω_{p+}^u from the original Markov chain. Following the same notation, we denote by $\Omega_{t,r}^u$ the set of the allowed locations at time t.

5.4.4 Adversary Model

We assume a powerful adversary that knows the user Markov chain $\mathbb{M}^u$. In addition, to incorporate side channel information, we further assume that the adversary can construct a larger Markov chain $\mathbb{M}^a = (\Omega^a, P^a)$ by merging different

behaviors (Markov chains) from other users as well, i.e. $\mathbb{M}^u \subseteq \mathbb{M}^a$. For example, if $\mathbb{M}^u$ represents the routes taken by user u, our model allow for the adversary to have additional knowledge about alternate routes taken by other users. The adversary can use these additional information to infer more about the user u.

The Markovian adversary utilizes $\mathbb{M}^a$ along with the released context state $\tilde{x}_t^u$ to infer the current user state. We denote by $\widehat{X}_t^a$ the random variable representing the adversarial belief about the actual user state. In other words, the adversary maintains a prior belief about X_t^u being a sensitive state, denoted by $\mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p}^u)$. Once a new state $\tilde{x}_t^u$ is released, the adversary uses it to update the posterior belief denoted by $\mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p}^u \mid \widetilde{X}_t^u = \tilde{x}_t^u)$.

The Markovian assumption on the adversary also implies the following independence and reverse-time independence properties:

$$\begin{aligned}
&\mathbb{P}(\widehat{X}_t^a = \hat{x}_t^u \mid \widehat{X}_1^a = \hat{x}_1^u, \ldots, \widehat{X}_{t-1}^a = \hat{x}_{t-1}^u, \widetilde{X}_1^u = \tilde{x}_1^u, \widetilde{X}_t^u = \tilde{x}_t^u) \\
&\qquad = \mathbb{P}(\widehat{X}_t^a = \hat{x}_t^u \mid \widehat{X}_{t-1}^a = \hat{x}_{t-1}^u, \widetilde{X}_t^u = \tilde{x}_t^u), \qquad (5.1) \\
&\mathbb{P}(\widehat{X}_t^a = \hat{x}_t^u \mid \widehat{X}_{t+1}^a = \hat{x}_{t+1}^u, \ldots, \widehat{X}_T^a = \hat{x}_T^u, \widetilde{X}_{t+1}^u = \tilde{x}_{t+1}^u, \widetilde{X}_T^u = \tilde{x}_T^u) \\
&\qquad = \mathbb{P}(\widehat{X}_t^a = \hat{x}_t^u \mid \widehat{X}_{t+1}^a = \hat{x}_{t+1}^u, \widetilde{X}_{t+1}^u = \tilde{x}_{t+1}^u). \qquad (5.2)
\end{aligned}$$

The above two properties imply that a future decision of the adversary is independent of the past decisions given the current state of the Markov chain. Furthermore, if the adversary tries to use future information to update a past decision then the past decision is again independent of the future information given the current state.

In addition, the following refer to the inference capabilities of the adversary:

$$\mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p}^u \mid \widetilde{X}_t^u \in \Omega_{t,p+}^u) = 1 \qquad (5.3)$$

$$\mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p}^u \mid P_{t,r}^u(\widetilde{X}_{t-1}^u, \widetilde{X}_t^u) = 0) = 1 \qquad (5.4)$$

The first assumption (Eqn. (5.3)) limits the system from releasing states that lead to sensitive states. The second assumption (Eqn. (5.4)) prevents the system from

releasing states that are not compatible with the Markov chain. The above model implies that the adversary gains a lot of information about the sensitive state whenever the released states do not obey the Markov chain and/or the sensitive state is actually released.

5.4.5 Metrics

Let $\mathbb{X}^u = \{start, x_1^u, \ldots, x_T^u, end\},\ x_t^u \in \Omega_t^u, t \in \mathbb{T}$, be the actual user trajectory over the duration of day. Similarly, $\widetilde{\mathbb{X}}^u = \{start, \tilde{x}_1^u, \ldots, \tilde{x}_T^u, end\}$, $\tilde{x}_t^u \in \Omega_{t,r}^u, t \in \mathbb{T}$, are location released over the day (note that the adversary sees only sensor data corresponding to the released contexts).

There are three considerations in generating $\widetilde{\mathbb{X}}^u$. First, the path should be as close as possible (minimum distortion) to the actual trajectory $\mathbb{X}^u$. We denote this metric as Ψ_{util}. Second, the path should be *typical*, i.e., $\widetilde{\mathbb{X}}^u$ should respect the Markov chain model, to ensure plausibility. We denote this metric as Ψ_{plau}. Finally, the path should provide privacy using metric Ψ_{priv} to the user. We formally define all the three metrics below.

Utility: This metric is designed to capture the mismatch (distortion) between the actual and released trajectories. Consider the actual user location at time t, denoted by x_t^u and the released location $\widetilde{x}_t^u$. Define the utility function:

$$\mathbb{F}_{util} : \Omega_t^u \times \Omega_{t,r}^u \rightarrow [0, 1]$$

This function takes as an arguments both x_t^u and $\widetilde{x}_t^u$ and return a value between 0 and 1. The utility function should return maximum value whenever the released location is equal to the actual location, i.e. $\mathbb{F}(x_t^u, \widetilde{x}_t^u) = 1$ whenever $x_t^u = \widetilde{x}_t^u$.

Now, we can define our utility metric Ψ_{util} as:

$$\Psi_{util} = \frac{1}{|\mathbb{T} \setminus \mathbb{T}_p|} \sum_{\substack{t \in \mathbb{T} \setminus \mathbb{T}_p \\ x_t^u \in \mathbb{X}^u, \tilde{x}_t^u \in \widetilde{\mathbb{X}}^u}} \mathbb{F}_{util}(x_t^u, \tilde{x}_t^u), \tag{5.5}$$

where $\mathbb{T}_p = \{t \mid x_t^u \in \Omega_{t,p+}^u\}$ is time instances when the user is in a sensitive location.

In Eqn. (5.5), we calculate the overall utility gain. By maximizing this quantity, we minimize the output distortion. We ignore the times when the user was in a sensitive state. Loss in utility due to the sensitive locations is an inevitable consequence of protecting privacy.

Plausibility: This metric is designed to ensure that the sequence of released locations in $\widetilde{\mathbb{X}}^u$ is typical as per the residual Markov chain $\mathbb{M}_r^u$. In other words, the probability of the chosen trajectory should be high enough to make it seem plausible to an adversary.

We define Ψ_{plau} as the sum of the probabilities of the released path over each time instance t normalized by the path with the maximum probabilities:

$$\Psi_{plau} = \frac{\sum_{t\in\mathbb{T}, \tilde{x}_{t-1}^u, \tilde{x}_t^u \in \widetilde{\mathbb{X}}^u} P_{t,r}^u(\tilde{x}_{t-1}^u, \tilde{x}_t^u)}{\sum_{t\in\mathbb{T}} \max_{\tilde{x}_{t-1}^u, \tilde{x}_t^u \in \Omega_{t,r}^u} P_{t,r}^u(\tilde{x}_{t-1}^u, \tilde{x}_t^u)}. \tag{5.6}$$

Privacy: The adversary maintains a prior belief, $\mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p}^u)$ on the user being in a private state at time t. On observing the released states the adversary revises his belief on the current state and computes the posterior probability. We define the privacy metric Ψ_{priv} over the entire released trajectory as:

$$\Psi_{priv} = \sum_{t\in\mathbb{T}} \mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p}^u \mid \widetilde{\mathbb{X}}^u) - \mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p}^u). \tag{5.7}$$

Eqn. (5.7) can be interpreted as the information gain after observing the released data. Thus, the posterior is always greater than or equal to the prior. Also, in Eqn. (5.7) we condition on the data released over the entire day $\widetilde{\mathbb{X}}^u$ protecting the user not only against data released until time t, but against all the data released over a day.

5.4.6 Problem Statement

Using the above definitions and notation, we can state the problem as follows.

Problem 7. *The privacy-utility tradeoff can be formulated as the following optimization problem:*

$$\max \qquad \gamma\Psi_{util} + (1-\gamma)\Psi_{plau} \tag{5.8}$$

$$s.t. \qquad \Psi_{priv} = 0 \tag{5.9}$$

where γ is a tuning parameter such that $0 \leq \gamma \leq 1$.

Note that setting $\gamma = 0$, is discouraged as it may lead for unacceptable performance by not penalizing the deviation between the paths $\mathbb{X}^u$ and $\widetilde{\mathbb{X}}^u$. Also note that both Ψ_{util} and Ψ_{plau} are normalized to a value between zero and one. Maximizing the objective in Eqn. (5.8) ensures that the released context states and the corresponding sensor data released closely match the actual data, and are plausible in the case of falsification. The parameter γ can be used to prioritize between the utility and the plausibility metrics. The constraint in Eqn. (5.9) ensures that the adversary does not learn anything more about the actual context state at any time $t \in \mathbb{T}$ using the released sensor data (zero privacy leakage).

5.5 Algorithmic Analysis of Blocks

In this section, we detail the algorithms used in the building blocks of iDeceit.

5.5.1 Trusted Server

As discussed before, iDeceit uses a trusted server to train the Markov chain based models. A user uploads a mobility trace and the server computes the model with locations as nodes and transition probabilities between nodes determined by the frequency of travel between the corresponding locations. The server also

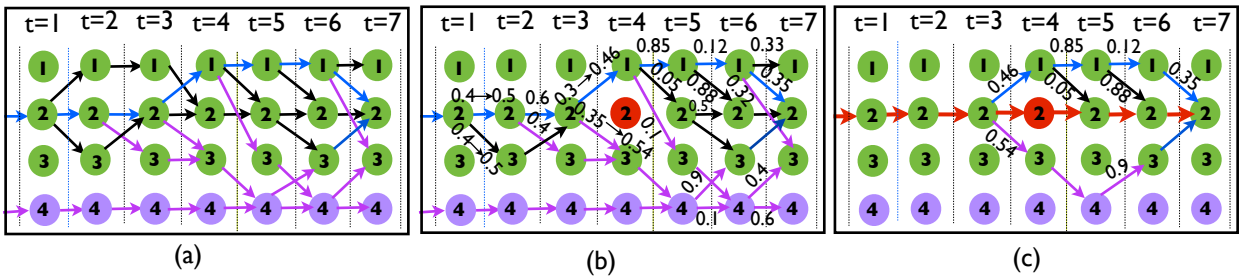


Figure 5.3: Illustration of the steps involved in generating a falsified trajectory. (a) A super Markov chain is shown where the nodes in green are from the user Markov chain and the additional nodes in violet are from the remaining population. Edges in black are ones whose transition probability is same as that in the user Markov chain, ones in blue have different probability than in the user Markov chain, and violet edges are absent in the user Markov chain. (b) Residual Markov chain with state $\omega^u_{2,4}$ as the private state. (c) Actual user trajectory is shown by the red arrows.

utilizes information collected from different users in order to extend the search space. By aggregating location information from all users (Ω_s), along with the transition probabilities from all users (P_s), we can construct the *super* Markov chain $\mathbb{M}_s = (\Omega_s, P_s)$.

An example of a super Markov chain is shown in Figure 5.3(a). The nodes in green are due to the user and the ones in violet are from the remaining population (excluding the user). There are three kinds of edges in $\mathbb{M}_s$. The edges in black are ones that occur only in the user Markov chain. The ones in blue are common to both the super and user Markov chains with possibly different probabilities in each. Finally, the ones marked violet are exclusively due to the super Markov chain.

5.5.2 Privacy-Aware Model Generator (PAG)

The super Markov chain contains all the possible locations and transitions for all the users. However, for effective falsification, we want to select nodes and edges

that strongly reflect user specific behavior. To achieve this we generate a projected Markov chain, $\mathbb{M}^u_\pi = (\Omega^u_\pi, P^u_\pi)$ as follows.

Consider two locations x^u_{t-1} and x^u_t such that a blue edge exist between them in the super Markov chain $\mathbb{M}_s = (\Omega_s, P_s)$. Then transition probabilities on all the outgoing edges from state $x^u_{t-1} \in \Omega_s$ are modified as follows:

$$P^u_{t,\pi}(x^u_{t-1}, y_t) = \alpha P^u_t(x^u_{t-1}, y_t) + (1-\alpha) P_{t,s}(x^u_{t-1}, y_t) \tag{5.10}$$

where $0 \leq \alpha \leq 1$ and $y_t \in \Omega_{t,s}$. If $y_t \notin \Omega^u_t$ then we set $P^u_t(x^u_{t-1}, y_t)$ to zero. For example, in Figure 5.3(a), there is a blue edge between $\omega^u_{2,3}$, and $\omega^u_{1,4}$. Using Eqn. (5.10), the transition probabilities $P_{t,s}(\omega^u_{2,3}, \omega^u_{1,4})$, $P_{t,s}(\omega^u_{2,3}, \omega^u_{2,4})$ and $P_{t,s}(\omega^u_{2,3}, \omega^u_{3,4})$ are adjusted in the projected Markov chain $\mathbb{M}^u_\pi$.

The final step is to remove all the sensitive locations Ω^u_{p+}, along with the associated transitions, from the projected Markov chain. We call the resulting graph as the *residual* (allowed) Markov chain $\mathbb{M}^r_u = (\Omega^u_r, P^u_r)$. That is,

$$\Omega^u_r = \Omega^u_\pi \setminus \Omega^u_{p+}, \quad P^u_r = P^u_\pi.$$

A residual graph is shown in Figure 5.3(b) with the set of sensitive state $\Omega^u_p = \{\omega^u_{2,4}\}$. We set $P^u_{5,r}(\omega^u_{2,4}, \omega^u_{2,5}) = 0$. Now, $P^u_{4,r}(\omega^u_{1,3}, \omega^u_{2,4}) = 1$ therefore we set $P^u_{4,r}(\omega^u_{1,3}, \omega^u_{2,4}) = 0$ and since its outdegree is zero we push $\omega^u_{1,3}$ as a private node and similarly for node $\omega^u_{1,2}$. After setting $P^u_{2,r}(\omega^u_{2,1}, \omega^u_{1,2}) = 0$, the deleted probability of 0.2 is proportionately divided among the remaining edges $(\omega^u_{2,1}, \omega^u_{2,2})$ and $(\omega^u_{2,1}, \omega^u_{3,2})$. The same is done for node $\omega^u_{2,3}$. Edges marked as $p_1 \to p_2$ indicate the probabilities before (p_1) and after (p_2) adjustment.

Finally, by construction, the *residual* Markov chain removes the edges connecting every sensitive node in Ω^u_{p+}. Thus, there exist no path in the residual graph that contain a sensitive node ensuring zero-loss privacy for any released path.

5.5.3 Path Planner (PP)

The PP block takes as input the (i) user Markov chain, $\mathbb{M}^u$ (ii) residual Markov chain, $\mathbb{M}_r^u$ (iii) augmented sensitive locations Ω_{p+}^u (iv) user's current location ($x_t^u \in \Omega_t^u$) and (v) the last released location $\tilde{x}_{t-1}^u \in \Omega_{t-1,r}^u$. It calculates the next released location $\tilde{x}_t^u \in \Omega_{t,r}^u$.

5.5.3.1 PathPlan Algorithm

The PathPlan (see Algorithm 1) outlines the steps involved in generating the next safe location which is to be released. If the current location x_t^u is sensitive, the $getNextSafeState$ is called to get the next safe location (line 3). If the current location is safe and there exists an edge from the previous released location $\tilde{x}_{t-1}^u$ to the current location, then release the current location (line 5). Otherwise, the current location is safe, but there exist no transition (because of falsification at an earlier time), again the $getNextSafe-$
$State$ is invoked (line 7).

There are two parts to the $getNextSafeState$ function: (i) Prediction of the next safe state (ii) Finding a path that converges to the predicted safe state. We discuss them below.

5.5.3.2 Prediction

Although the system maintains prior knowledge about the user behavior, captured by $\mathbb{M}^u$, the exact location at run-time depends on the user decisions and is reflected in the current state x_t^u. To cope with the unpredictability in the users' current location the PathPlan algorithm is executed in the receding horizon sense. At every time instant t, depending on the current location x_t^u, we need to predict the most likely next safe location the user will visit and plan a path to it. This process is then repeated at each time step based on the actual value of x_t^u.

Algorithm 1 PathPlan: Safe Context State Generator

Require: $\Omega^u_{p+}, \mathbb{M}^u, \mathbb{M}^u_r, x^u_t, \tilde{x}^u_{t-1}$

1: **for** $t \in \mathbb{T}$ **do**
2: **if** $x^u_t \in \Omega^u_{t,p+}$ **then**
3: $\tilde{x}^u_t \leftarrow getNextSafeState(\tilde{x}^u_{t-1}, x^u_t, \mathbb{M}^u, \mathbb{M}^u_r)$
4: **else if** $P^u_{t,r}(\tilde{x}^u_{t-1}, x^u_t) > 0$ **then**
5: $\tilde{x}^u_t \leftarrow x^u_t$
6: **else**
7: $\tilde{x}^u_t \leftarrow getNextSafeState(\tilde{x}^u_{t-1}, x^u_t, \mathbb{M}^u, \mathbb{M}^u_r)$
8: **end if**
9: **end for**

For prediction, we use the probabilities of the paths in $\mathbb{M}^u$. Starting from the current user location x^u_t, we select the most probable next safe location by traversing through the edges corresponding to the highest probabilities. Algorithm 2 formalizes this idea in the steps of the Predict algorithm. For a certain look ahead window length L, the Predict algorithm can return an empty solution which implies that the user is expected to still be in a sensitive location up to time $t+L$. Thus, we need to increase L and re-invoke the Predict algorithm and continue to do so until we find a safe location. In the worst case this procedure will terminate at $L = T - t + 1$ corresponding to the *end* state. Going back to Figure 5.3(c), at time $t = 4$ and $L = 1$ the next safe location predicted is $\omega^u_{2,5}$.

Algorithm 2 Predict: Predict the Safe State at time t+L

Require: $\mathbb{M}^u, x^u_t, L$

1: $x_{pred} = x^u_t$
2: **for** $l \in \{1, \ldots L\}$ **do**
3: $x_{pred} \leftarrow \arg\max_{x^u_{t+l} \in \Omega^u_{t+l}} P^u_{t+l}(x_{pred}, x^u_{t+l})$
4: **end for**

5.5.3.3 Path Finding

The next step is to find an optimum path that connects the current location x_t^u to the next most probable location state x_{pred}. Optimality of the path is measured using the utility metric in Eqn. (5.5) and the plausibility metric in Eqn. (5.6). Path selection is always performed from among the paths in $\mathbb{M}_r^u$. This follows from the fact that there are many more paths to choose from in $\mathbb{M}_r^u$ than in $\mathbb{M}^u$ and all those paths are safe by construction.

As shown in Figure 5.3(c), there are four different paths that can be taken from $\omega_{2,3}^u$ to converge to the user trajectory. The possible paths are: $Path_1 = (\omega_{2,3}^u \rightarrow \omega_{1,4}^u \rightarrow \omega_{2,5}^u)$, $Path_2 = (\omega_{2,3}^u \rightarrow \omega_{1,4}^u \rightarrow \omega_{1,5}^u \rightarrow \omega_{1,6}^u \rightarrow \omega_{2,7}^u)$, $Path_3 = (\omega_{2,3}^u \rightarrow \omega_{3,4}^u \rightarrow \omega_{4,5}^u \rightarrow \omega_{3,6}^u \rightarrow \omega_{2,7}^u)$ and $Path_4 = (\omega_{2,3}^u \rightarrow \omega_{1,4}^u \rightarrow \omega_{1,5}^u \rightarrow \omega_{2,6}^u)$. We start by presenting algorithm choices which choose between these different paths and in the process maximize one or the other metric in Eqn. (5.8). Finally, we will also provide a technique to jointly maximize the two metrics.

- **Shortest Path (SP):** We start with the extreme case when the tuning parameter in Problem 7, γ, is set to one. Hence, we focus only on the maximization of Ψ_{util}. It is straightforward to show that maximizing Ψ_{util} is equivalent to finding the shortest path between the prior released location $\tilde{x}_{t-1}^u$ and the next safe location x_{pred}. Standard techniques like Dijkstra's algorithm can be used to find such a path. Since we focus only on maximizing Ψ_{util}, we can ignore the probabilities associated with the different paths and treat all edges as having equal weights. The SP algorithm iterates over different look ahead window sizes until the Predict algorithm finds a safe location and there also exists a path to it. These steps are illustrated in Algorithm 3.

  In Table 5.1, SP algorithm chooses $Path_1$ requiring two hops to converge to the next safe location $\omega_{2,5}^u$.

- **Highest Probability Edge (HiPE):** We consider the other extreme by set-

Algorithm 3 SP: Shortest Path algorithm

Require: $\mathbb{M}^u, \mathbb{M}^u_r, x_t, \tilde{x}^u_{t-1}$

for $L \in \{t, \dots T\}$ **do**
  $x_{pred} \leftarrow \text{Predict}(\mathbb{M}^u, x^u_t, L)$
  if $x_{pred} \neq \emptyset$ **then**
    $path \leftarrow \text{Dijkstra}(\tilde{x}^u_{t-1}, x_{pred}, \Omega^u_r)$
    if $path \neq \emptyset$ **then**
      Return $path$ and exit.
    end if
  end if
end for

ting the tuning parameter in Problem 7, γ, to zero and hence we focus on maximizing only Ψ_{plau}. A greedy approach to maximizing Ψ_{plau} is to always traverse through the edges with the highest transition probability. However, the problem occurs when the user moves into a low probability safe location. At that point the released path diverges and can over time continue to diverge from the user trajectory leading to loss in utility. Accordingly, the case where $\gamma = 0$ is prohibited in the first place for Problem 7. However, this discussion provides insight for the next algorithm. In Table 5.1, for the specific set of edge probabilities, $Path_3$, chosen using this approach, converges to the actual user trajectory.

- **Maximize Utility and Plausibility (MUP):** In this algorithm we consider the joint optimization of Ψ_{util} and Ψ_{plau}. Algorithm 4 outlines the steps involved in the MUP algorithm. Selecting the shortest path between $\tilde{x}^u_{t-1}$ and x_{pred} will ensure good utility. However, to account for plausibility, we apply Dijkstra's algorithm taking the probabilities associated with edges as weights, i.e. for an edge e with probability p, we assign a weight equal to $1 - p$. Hence, Dijkstra's algorithm is going to minimize over $1 - p$ which is equivalent to maximizing

the plausibility metric. This fact is reflected in the MUP algorithm by invoking Dijkstra's algorithm with $\mathbb{M}_r^u$ as an argument (line 4 in Algorithm 4) compared to the SP algorithm where it is invoked with Ω_r^u (line 4 in Algorithm 3) which does not reflect the probabilities associated with edges.

Unlike the SP algorithm, MUP does not terminate once a path is found. Instead it explores different paths over the entire look ahead window, associating with each path a cost measured by Eqn.(5.8). The path with the maximum cost is selected. In Table 5.1, the cost for different paths using MUP are shown and $Path_4$ with maximum cost is selected.

Algorithm 4 MUP: Maximize Utility and Plausibility algorithm

Require: $\mathbb{M}^u, \mathbb{M}_r^u, x_t, \tilde{x}_{t-1}^u$

1: **for** $L \in \{t, \ldots T\}$ **do**
2: $x_{pred} \leftarrow \text{Predict}(\mathbb{M}^u, x_t^u, L)$
3: **if** $x_{pred} \neq \emptyset$ **then**
4: $path_L \leftarrow \text{Dijkstra}(\tilde{x}_{t-1}^u, x_{pred}, \mathbb{M}_r^u)$
5: **if** $path_L \neq \emptyset$ **then**
6: $cost_L \leftarrow computeCost(path_i)$
7: **end if**
8: **end if**
9: **end for**
10: $L_{\text{max cost}} \leftarrow \arg\max_{L \in \{t, \ldots T\}} cost_L$
11: $path \leftarrow path_{L_{\text{max cost}}}$

5.6 Theoretical Guarantees

In this section, we present our theoretical result which shows the falsification of the location leads to zero-loss privacy guarantee.

Algorithm Used	$Path_1$	$Path_2$	$Path_3$	$Path_4$
SP	**1**	0.67	0.67	0.833
HiPE	x	x	**selected**	x
MUP	0.897	0.684	0.867	**0.905**

Table 5.1: Convergence cost to the actual user trajectory in Figure 5.3(c) for different algorithm choices.

Definition 8. *A vertex cut $\mathcal{C}$ of a graph is defined as the set of vertices whose removal renders the graph disconnected.*

Definition 9. *The vertex connectivity κ of a graph is defined as the size of the minimal vertex cut, i.e. $\kappa = \min |\mathcal{C}|$.*

Theorem 10. *Let κ_r^u denote the vertex connectivity of the graph representing the Markov chain M_r^u. If the following condition is satisfied*

$$\kappa_r^u > \max_t |\Omega_{t,p+}^u|,$$

then iDeceit ensures zero-loss privacy, i.e.

$$\Psi_{priv} = \sum_{t \in \mathbb{T}} \mathbb{P}(\widehat{X}_t^a \in \Omega_{p,t}^u \mid \widetilde{\mathbb{X}}) - \mathbb{P}(\widehat{X}_t^a \in \Omega_{p,t}^u) = 0.$$

Proof Of Theorem 10

First, we study how the posterior beliefs of the adversary changes by observing only the current released state. Define the set:

$$\mathbb{S}_{t,r}^u(\hat{x}_{t-1}^a) = \{\tilde{x}_t^u \in \Omega_{t,r}^u \setminus \Omega_{t,p+}^u \mid P_{t,r}^u(\hat{x}_{t-1}^a, \tilde{x}_t^u) > 0\}.$$

The set $\mathbb{S}_{t,r}^u(\hat{x}_{t-1}^a)$ represents the safe states at time t which have edges from the states in the previous time stage $t-1$. We can now show the first result as follows:

Proposition 11. ***If*** *the following conditions are satisfied:*

1. *the set of allowed safe states at time t, denoted by $\mathbb{S}^u_{t,r}(\hat{x}^a_{t-1})$, is non-empty,*

2. $\mathbb{P}(\widetilde{X}^u_t \in \mathbb{S}^u_{t,r}(\hat{x}^a_{t-1})) = 1$,

Then *the posterior knowledge of the adversary does not change by observing the released the value of $\widetilde{X}^u_t$, i.e.:*

$$\mathbb{P}(\widehat{X}^a_t \in \Omega^u_{t,p+} \mid \widetilde{X}^u_t \in \mathbb{S}^u_{t,r}(\hat{x}^a_{t-1})) = \mathbb{P}(\widehat{X}^a_t \in \Omega^u_{t,p+}).$$

Proof. We first note that all the states in the set $\mathbb{S}^u_{t,r}(\hat{x}^a_{t-1})$ does match any of the constraints in the adversary model (Eqn. (5.3) and Eqn. (5.4)). To proof the result, we resort to Bayes rule as follows:

$$\begin{aligned}
&\mathbb{P}(\widehat{X}^a_t \in \Omega^u_{t,p+} \mid \widetilde{X}^u_t \in \mathbb{S}^u_{t,r}(\hat{x}^a_{t-1})) \\
&= \frac{\mathbb{P}(\widetilde{X}^u_t \in \mathbb{S}^u_{t,r}(\hat{x}^a_{t-1}) \mid \widehat{X}^a_t \in \Omega^u_{t,p+})\mathbb{P}(\widehat{X}^a_t \in \Omega^u_{t,p+})}{\mathbb{P}(\widetilde{X}^u_t \in \mathbb{S}^u_{t,r}(\hat{x}^a_{t-1}))} \\
&= \frac{1 \times \mathbb{P}(\widehat{X}^a_t \in \Omega^u_{t,p+})}{1} = \mathbb{P}(\widehat{X}^a_t \in \Omega^u_{t,p+}).
\end{aligned}$$

□

We are also interested in studying how the adversary knowledge will change if the adversary is using information from time $t+1$ to infer about the state at time t. In other words, how the posterior knowledge changes if the adversary observed future information and moves backward in time to update his knowledge about the state at time t. This can be stated as follows:

Proposition 12. ***If*** *the following conditions are satisfied:*

1. *the set of allowed safe states at time $t+1$, denoted by $\mathbb{S}^u_{t+1,r}(\hat{x}^a_t)$, is non-empty,*

2. $\mathbb{P}(\widetilde{X}^u_{t+1} \in \mathbb{S}^u_{t+1,r}(\hat{x}^a_t)) = 1$,

Then *the posterior knowledge of the adversary does not change by observing the released the value of* $\widetilde{X}_t^u$*, i.e.:*

$$\mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p+}^u \mid \widetilde{X}_{t+1}^u \in \mathbb{S}_{t+1,r}^u(\hat{x}_t^a)) = \mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p+}^u).$$

Proof. The proof follows the same steps of Proposition 12. □

We then extend these results to study how the belief of the adversary changes by monitoring the whole released trajectory $\widetilde{\mathbb{X}}^u$ as follows:

Lemma 13. ***If*** *the following conditions are satisfied:*

1. *the set of allowed safe states at times* t *and* $t+1$*, denoted by* $\mathbb{S}_{t,r}^u(\hat{x}_{t-1}^a)$*,* $\mathbb{S}_{t+1,r}^u(\hat{x}_t^a)$*, are non-empty,*
2. $\mathbb{P}(\widetilde{X}_t^u \in \mathbb{S}_{t,r}^u(\hat{x}_{t-1}^a)) = 1$*,*
3. $\mathbb{P}(\widetilde{X}_{t+1}^u \in \mathbb{S}_{t+1,r}^u(\hat{x}_t^a)) = 1$*,*

Then *the posterior knowledge of the adversary does not change by observing the released trajectory* $\widetilde{\mathbb{X}}^u = \{start, \widetilde{X}_1^u, \ldots, \widetilde{X}_T^u, end\}$*, i.e.:*

$$\mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p+}^u \mid \widetilde{\mathbb{X}}^u) = \mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p+}^u).$$

Proof. It follows from the independence assumptions on the Markovian adversary (Eqn. (5.1) and Eqn. (5.2)) that

$$\mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p+}^u \mid \mathbb{X}^u) = \mathbb{P}(\widehat{X}_t^a \in \Omega_{t,p+}^u \mid \widetilde{X}_t^u, \widetilde{X}_{t+1}^u)$$

Combining the previous equality with Propositions 11 and 12, we conclude the result. □

Note that the previous results asks for the set $\mathbb{S}_{t,r}^u(\hat{x}_{t-1}^a)$ to be non-empty. It then important to study the conditions for which the non-emptiness condition on $\mathbb{S}_{t,r}^u(\hat{x}_{t-1}^a)$ is satisfied. This condition can be related to the graph properties of the Markov chain as follows.

Lemma 14. *Let κ_r^u denote the vertex connectivity of the graph representing the Markov chain M_r^u. If the following condition is satisfied:*

$$\kappa_r^u > \max_t |\Omega_{t,p+}^u|, \tag{5.11}$$

then the set $\mathbb{S}_{t,r}^u(\hat{x}_{t-1}^a)$ is non-empty for all time $t \in \mathbb{T}$ and for all $\hat{x}_{t-1}^a \in \Omega_t^a$.

Proof. Assume, for the sake of contradiction, that the condition in (Eqn. (5.11)) is satisfied and for time t there exists $\hat{x}_{t-1}^a$ such that the set $\mathbb{S}_{t,r}^u(\hat{x}_{t-1}^a)$ is empty. However, it follows from the definition of the set $\mathbb{S}_{t,r}^u(\hat{x}_{t-1}^a)$ that $\mathbb{S}_{t,r}^u(\hat{x}_{t-1}^a)$ is empty if and only if after removing the nodes in $\Omega_{t,p+1}^u$, the graph become disconnected at time t. Hence, this implies that $\kappa_r^u \leq |\Omega_{t,p+}^u|$. Since t is arbitrary, we arrive at a contradiction. □

The previous two results paves the way to show the main privacy guarantee of iDeceit which can be stated as follows.

Proof. (Theorem 10) Follows directly by applying Lemma 13 and Lemma 14. □

5.7 Implementation

iDeceit provides prototype implementation of the following components (shown in green in Figure 5.4): (i) Context Engine (CE) (ii) Data Synthesis block (DS) (iii) PAG and Path Planner (iv) A data path for distributing the synthetic data to the apps. While (i) and (ii) are specific to location, (iii) and (iv) are generic implementations.

CE and DS blocks: iDeceit takes as input a dictionary, provided by the server. The dictionary is a mapping between location names configured by the user, actual GPS measurements in terms of latitude, longitude pairs, and a server generated label unique across different user traces. The Markov chains (user and super) are constructed using the unique labels as states. This dictionary acts

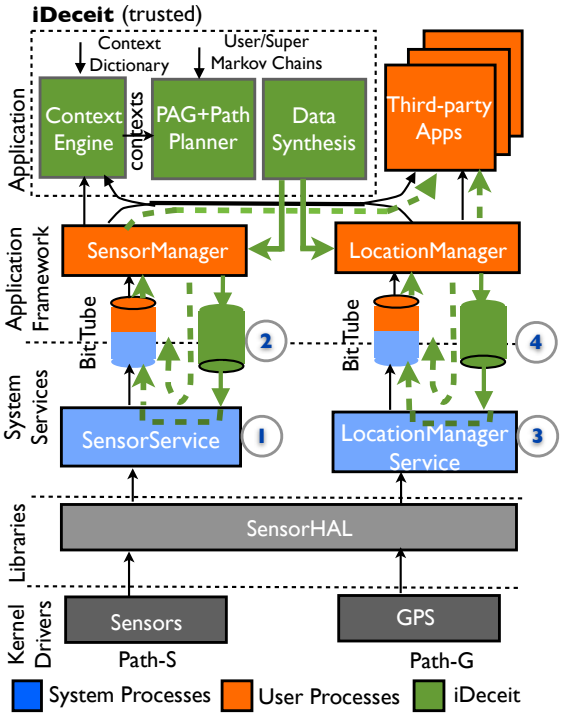


Figure 5.4: Android Implementation of iDeceit.

as a CE allowing reverse lookup from GPS measurements to locations, and from locations to the unique labels in the models.

The DS block outputs the GPS coordinates for the location output by the PP block. There exist a one-to-one mapping between a latitude and longitude pair and a location, hence the dictionary also doubles up as the DS block. For a given falsified location, the corresponding GPS measurement can be looked up in the dictionary and released. The feasibility of location lookup from GPS measurements at a large scale on mobile platforms has already been demonstrated in [GJP12].

PAG and the PP blocks: The PAG block takes as input sensitive locations configured by the user, maps them to labels in the super Markov chain using the dictionary and generates the residual Markov chain. Similarly, the PP block implements the MUP algorithm and outputs the location to the DS block.

Data Path: Two existing data paths in Android are shown in Figure 5.4. *Path-S* (without the iDeceit components in green) illustrate the flow of data from sensors like Accelerometer, Gyroscope, Light etc. to apps in Android. The corresponding path for the GPS data is shown by *Path-G*. Sensor data is continuously polled by the {`Sensor, LocationManager`}`Service` using the SensorHAL and passed to the corresponding {`Sensor, Location`}`Manager` instances. The managers are user-facing proxies to the services, and can be instantiated by the apps for accessing sensor data. On instantiation by an app, each manager initiates a `BitTube` connection with their corresponding service. `BitTube` provides a full-duplex socket pair. When data is received at the service it writes the data into one end of the socket pair and it is read by the manager at the other end and passed to the app. However, data in both the paths can currently flow in only one direction: from the sensors to the apps. Note, the color code for the existing Android components (excluding iDeceit components) represent process boundaries.

To push synthetic data to the apps, we need a $U-$shaped data path. The DS block uses the {`Sensor, Location`}`Manager` to push synthetic data to the {`Sensor, LocationManager`}`Service` and then using the original path the data can be sent to the apps (dotted green lines). To establish this path we need to make modifications to `SensorService`(①) and `BitTube`(②) in *Path-S* and in the `LocationManagerService`(③) and `BitTube`(④) in *Path-G*. We modified the `BitTube` structure to allow bidirectional communication by enabling write operation on the socket for the manager and read operation at the other end of the socket for the service on both paths.

Sensor events generated by the DS are populated in the `Sensor-Event` class used for sensor data and written to the socket. At {`Sensor, LocationManager`} `Service` we maintain circular buffers for each of the different types of sensors and their events. A separate thread is used to continuously poll the socket end point for synthetic data and populate the buffers. The user specified rules are written

to `/data/firewall-config` file. iDeceit notifies a service whenever the rule file is updated using Remote Procedure Call (RPC) (we implemented a `reloadConfig()` binder call for RPC). The rules specify the fully-qualified package name to identify the app, the Android sensor type (Android constants for identifying sensors) and source of synthetic data (replay, synthesize). For the replay mode, a trace location needs to be provided. The rules are read into a HashMap indexed using a combination of package name and sensor type. When the service needs to push data to apps, it checks if a rule exists for the given package name and sensor type and performs the required buffer switch. The default policy for playback is set to drop. Thus, if the hardware sensor generates an event and the circular buffer is empty, with playback configured for the sensor, the sensor data is dropped for that app, otherwise the data in the circular buffer is sent to the app. Our implementation is ongoing work and it does not support sending synthetic data at rates different than the one that is selected by the app. We ensure that only iDeceit is able to write data by comparing with the package name which is unique across Android.

5.8 Evaluation

We use data collected during a real-life mHealth study designed to provide insights into the relationship between physiological markers (stress, smoking etc.), physical activity and location (with IRB approval). We make use of only the location data collected as part of the study. We recruited 23 participants, of which 22 completed all 7 days of field study. Each participant carried a study phone to enable continuous data collection in their free living conditions, and during visits to significant places on each day during the study period. GPS data, sampled at 1 Hz, was continuously collected on the phone (avg. 10.82 hours of data per day per user) for a total of about 6 million data samples.

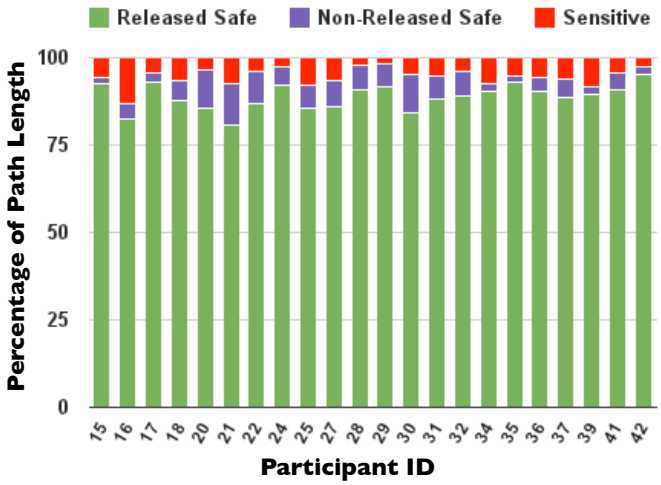


Figure 5.5: Average number of node types over a day for all users.

We performed preprocessing on the GPS data to identify significant places and associate semantic labels to those places. For each participant, spatial and temporal thresholds of 100m and 5min respectively, were used to find spatiotemporal clusters corresponding to significant places throughout the day. For clustering, we followed the algorithm proposed in [MBG13], resulting in an accuracy of 85.7%. Next, we assigned semantic labels to these places using the method proposed in [KR13]. We resolved any confusion in label assignment by manual inspection of the trace on a map. We ended up with six locations *Home, Work, University, Store, Restaurant, Other*. Missing locations were labeled as *Other*.

Setup for iDeceit: For model learning we use the leave-one-out cross validation. For each user, we repeated the experiment using data from each day as test data and the remaining days as training data. The training data was used to both learn the user Markov chain and together with data from all the other users used to construct the super Markov chain. The values reported in the plots are averaged over the different tests performed (seven for every user). We defined sensitive states in two steps. In the first step, we selected a time interval of the day (approx. 2 hour duration). In the second step, we selected a subset of the six places. We then defined the sensitive states to be the selected subset of places in

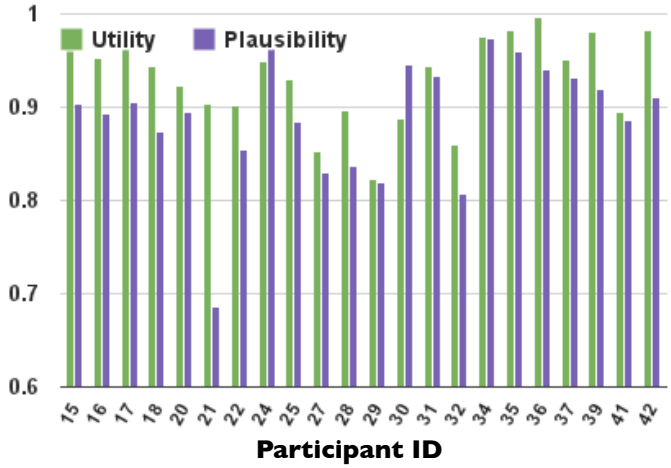


Figure 5.6: Max. utility (Ψ_{util} at $\gamma = 1$) and Max. plausibility (Ψ_{plau} at $\gamma = 0$) for all users.

the chosen interval. The residual Markov chain is generated based on the sensitive states. In a day, data is released every $5min$, i.e., $\mid \mathbb{T} \mid = 288$. For the utility metric in Eqn. (5.5), we consider $\mathbb{F}_{util}$ to be an indicator function $\mathbb{I}_{\{x_t^u = \tilde{x}_t^u\}}$ where x_t^u and $\tilde{x}_t^u$ are the actual and released locations respectively. The function takes a value of 1 when $x_t^u = \tilde{x}_t^u$ and 0 otherwise.

Node Distribution by Type: In Figure 5.5, a distribution of node types for each user is shown. Based on the chosen sensitive states, we observed that on an average $5\%(\approx 15 - 20$ out of 288) nodes in an actual trajectory are sensitive for a user on an day (shown in red) and needs to be falsified. A large fraction ($> 90\%$) of nodes (in green) are safe nodes and can be released. Only a small fraction of nodes ($< 5\%$) are safe but still need to be falsified (in violet). This is because either they lead to a sensitive state or to maintain plausibility of the released trajectory. The falsification of safe nodes contribute to the loss in utility.

Max. Utility and Max. Plausibility: For a given set of sensitive nodes, we execute the MUP algorithm by setting $\gamma = 1$ and find the maximum utility of the computed path. We then set $\gamma = 0$ and find the maximum plausibility of the newly computed path. Figure 5.6 shows the maximum utility and plausibility

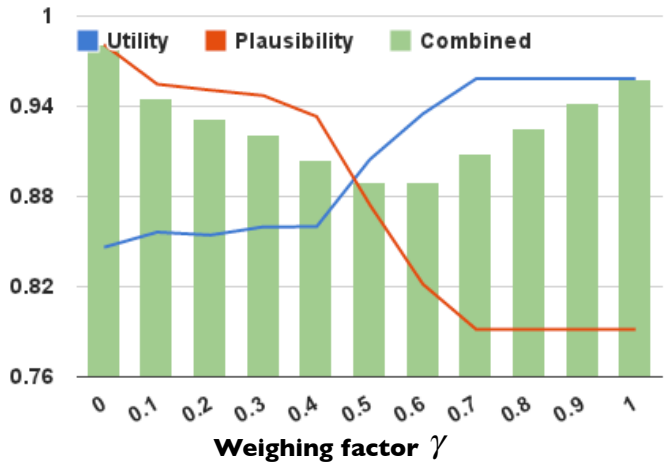


Figure 5.7: Variation in utility, plausibility and the combined (Eqn. (5.8)) metric for different values of γ.

values obtained for each user. We observe that although on average $\Psi_{util} \geq 0.90$ and $\Psi_{plau} \geq 0.80$, there are variations in these maximum values that we observe across users. This implies that using a suitable value of γ we can maximize them jointly for a given path.

Variation of Ψ_{util} and Ψ_{plau} with γ: We consider the variation in the objective function (Eqn. (5.8)) for different values of γ. We initialize γ to zero and increment it in steps of 0.1. For a value of γ we compute a path and obtain the utility (ψ_{util}), plausibility (ψ_{plau}) and the objective function for that path. We repeat this for each value of γ to obtain Figure 5.7. As expected, for small values of γ, the plausibility metric dominates and as the value of γ increases the utility factor starts to grow. They are almost equal for $\gamma = 0.5$. Hence, γ can be used to control between the two parameters.

Comparison Between SP, HiPE, MUP: We computed Ψ_{util} and Ψ_{plau} values for all the three algorithms, by setting $\gamma = 0.5$. In terms of Ψ_{util} only, SP performs better than MUP but in terms of Ψ_{plau}, MUP does better than SP. Similarly, for some participants $HiPE$ does better than MUP in terms of Ψ_{plau}. However, as shown in Figure 5.8, when we jointly optimize the two metrics, MUP

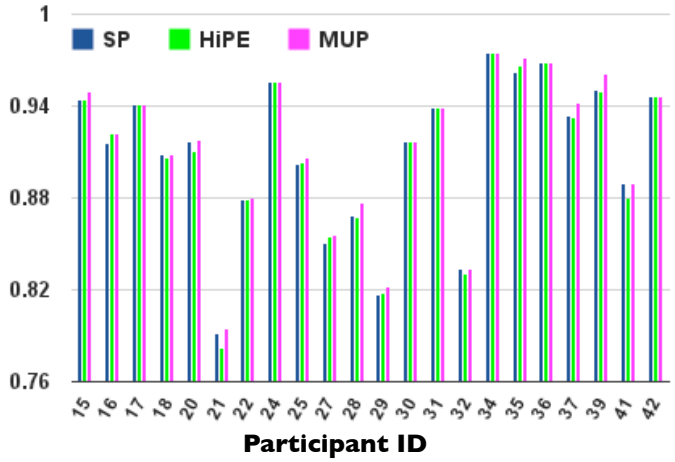


Figure 5.8: Comparison of three different algorithm in terms of the joint metric in Eqn. (5.8) where $\gamma = 0.5$.

always does better than both the algorithms.

Resource Consumption: We found that the *PathPlan* algorithm on an average took $\approx 2ms$ to execute on the phone, which is negligible compared to usual sampling rates on Android (SENSOR_DELAY_NORMAL around 10Hz, and SENSOR_DELAY_FASTEST around 200Hz). The memory overhead in storing the Markov chains is less than 1 MB. Also, note iDeceit runs when Playback is configured, and consumes negligible power. To further reduce power consumption pre-recorded traces maybe be replayed to the apps.

5.9 Related Work

Model-Based Privacy: We summarize other efforts towards model-based location privacy below. In [MRS09], mobility traces are used to compute pollution exposure. To protect path privacy a database of paths is maintained and a synthetic path which have the same pollution exposure as the original trace is released. However, in doing so, the utility from any service which relied on the trace is completely lost. We want to provide privacy while ensuring minimum distortion to

the original trace. Model-based privacy has also been proposed in [PCG13] where a DBN is used to infer the sensor combinations that can lead to sensitive contexts. The paper does not provide any formal analysis of privacy as the primary focus was to use the DBN to attain energy efficiency for continuous context computation. It shares contexts instead of sensor data and does not consider either falsification or synthesis of data. Similarly, in [GJP12], instead of GPS measurements, places are released to protect raw data. In [STT12, STL11] model-based obfuscation of instantaneous locations is considered. Obfuscation of traces through spatial and temporal cloaking is considered in [GG03]. A detailed exposition of other location privacy preserving techniques can be found in [Kru09]. The work that is closest to iDeceit is MaskIt [GNG12] and below we provide a detailed comparison.

Comparison With MaskIt: Consider the Markov chain shown in Figure 5.9. The possible locations are $C^u = \{1, 2, 3\}$. Incorporating time the state space $\Omega^u = \{\omega^u_{1,1}, \omega^u_{2,1}, \omega^u_{3,1}, \ldots, \omega^u_{3,3}, start, end\}$. The sensitive location specified by the user is $\Omega^u_p = \{\omega^u_{2,2}\}$. We have $P^u_1(start, 2) = 1$, $P^u_2(2, j) = \frac{1}{3}$, for all $j \in C^u$, $P^u_3(i, 2) = 1$, for all $i \in C^u$ and $P^u_4(2, end) = 1$. We show different cases and for each case compare the utility and privacy provided by our scheme to MaskIt. In MaskIt, the privacy algorithm has to choose between not releasing location data (which is called the suppress action and is denoted by $\perp$) or releasing it with certain probability. For sensitive locations, the suppress action $\perp$ is always chosen while for all other "safe" locations, the release probabilities are chosen such that Ψ_{priv} is always upper bounded by some user defined value. Note that unlike Problem 7, the privacy algorithm designed in MaskIt allows for non-zero privacy leakage.

Case 1: The actual user path is $\mathbb{X}^u = \{start, x^u_1 = \omega^u_{2,1}, x^u_2 = \omega^u_{2,2}, x^u_3 = \omega^u_{2,3}, end\}$. At times $t = 1$ and $t = 3$ the user is in a "safe" state and hence $\tilde{x}^u_1 = x^u_1$ and $\tilde{x}^u_3 = x^u_3$. At $t = 2$ the user is in a sensitive location. Accordingly, MaskIt will suppress $\omega^u_{2,2}$ with probability 1, i.e., $\mathbb{P}(\widetilde{X}^u_2 = \perp | \; X^u_2 = \omega^u_{2,2}) = 1$,

suppress $\omega^u_{1,2}$ with probability p_1 ($\mathbb{P}(\widetilde{X}^u_2 = \perp |\ X^u_2 = \omega^u_{1,2}) = p_1$), and $\omega^u_{3,2}$ with probability p_3 ($\mathbb{P}(\widetilde{X}^u_2 = \perp |\ X^u_2 = \omega^u_{3,2}) = p_3$).

The adversarial posterior knowledge is then given by $\mathbb{P}(\widehat{X}^a_2 = \omega^u_{2,2} \mid \widetilde{X}^u_2 = \perp)$ which can be computed as follows

$$\mathbb{P}(\widehat{X}^a_2 = \omega^u_{2,2} \mid \widetilde{X}^u_2 = \perp) = \frac{\mathbb{P}(\widetilde{X}^u_2 = \perp |\ \widehat{X}^u_2 = \omega^u_{2,2})\mathbb{P}(\widehat{X}^a_2 = \omega^u_{2,2})}{\mathbb{P}(\widetilde{X}^u_2 = \perp)}$$

where $P(\widetilde{X}^u_2 = \perp |\ \widehat{X}^a_2 = \omega^u_{2,2}) = 1$. This follows from the design of MaskIt where a location is suppressed whenever the adversary is going to infer that the current location is sensitive. Hence

$$\begin{aligned}
&\mathbb{P}(\widehat{X}^a_2 = \omega^u_{2,2} \mid \widetilde{X}^u_2 = \perp) \\
&= \frac{1 \times \mathbb{P}(\widehat{X}^a_2 = \omega^u_{2,2})}{\sum_{i=1}^{3} \mathbb{P}(\widetilde{X}^u_2 = \perp |\ X^u_2 = \omega^u_{i,2})\mathbb{P}(X^u_2 = \omega^u_{i,2})} \\
&= \frac{1}{\frac{1}{3}(1 + p_1 + p_3)} \mathbb{P}(\widehat{X}^a_2 = \omega^u_{2,2})
\end{aligned}$$

For MaskIt to achieve the privacy bound in Problem 7 (i.e. $\Psi_{priv} = 0$) p_1 and p_3 must be equal to 1, which in turn implies that MaskIt will completely suppress all outputs at time $t = 2$ regardless of whether x^u_2 is safe or not. Unlike MaskIt, iDeceit is going to falsify the data and release either $\omega^u_{1,2}$ or $\omega^u_{3,2}$. Thus, for this scenario, when the user is actually in a sensitive location, for achieving the privacy bound in Eqn. (5.7), iDeceit provides the same utility as MaskIt, but is better in terms of plausibility.

We now illustrate the working of iDeceit for the above scenario. At time $t = 2$, iDeceit is obliged to release a location. To protect against the adversary model previously discussed, iDeceit does not release the sensitive location $\omega^u_{2,2}$ or any other location that could lead to the sensitive state. Hence, the only states that can be released are $\omega^u_{1,2}$ and $\omega^u_{3,2}$, i.e., $\mathbb{P}(\widetilde{X}^u_2 \in \{\omega^u_{1,2}, \omega^u_{3,2}\}) = 1$. The adversarial

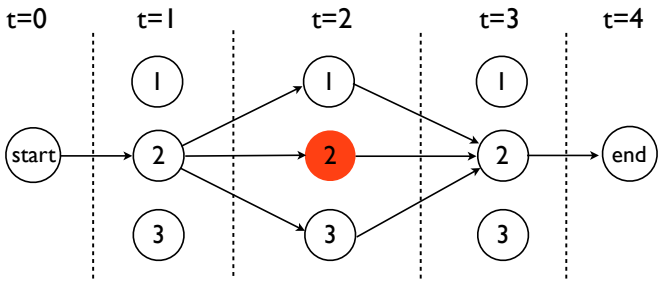


Figure 5.9: Markov chain between various context states. State $\omega^u_{2,2}$ is selected as sensitive by the user.

posterior knowledge is then given by:

$$\begin{aligned}
&\mathbb{P}(\widehat{X}^a_2 = \omega^u_{2,2} \mid \widetilde{X}^u_2 \in \{\omega^u_{1,2}, \omega^u_{3,2}\}) \\
&= \frac{\mathbb{P}(\widetilde{X}^u_2 \in \{\omega^u_{1,2}, \omega^u_{3,2}\} \mid \widehat{X}^a_2 = \omega^u_{2,2})\mathbb{P}(\widehat{X}^a_2 = \omega^u_{2,2})}{\mathbb{P}(\widetilde{X}^u_2 = \in \{\omega^u_{1,2}, \omega^u_{3,2}\})} \\
&= \frac{1 \times P(\widehat{X}^a_2 = \omega^u_{2,2})}{1} = \mathbb{P}(\widehat{X}^a_2 = \omega^u_{2,2}),
\end{aligned}$$

which ensures zero-loss privacy.

Case 2: User is in a safe state at $t = 2$ i.e., $x^u_2 \in \{\omega^u_{1,2}, \omega^u_{3,2}\}$ (actual user trajectory does not involve a sensitive location). Note that according to the Markov chain model, this occurs with higher probability than the user visiting a sensitive location. MaskIt will continue to suppress (with probabilities p_1 or p_2) the safe states at $t = 2$. However, iDeceit will release the actual state and thus outperform MaskIt in terms of both utility and plausibility.

MaskIt can enhance its utility by reducing p_1 and p_2, but it can do so only after tolerating a non-zero privacy leakage (definition of $\delta-$privacy in [GNG12]). But, iDeceit can achieve the same utility with zero privacy leakage. Most importantly, using the plausibility metric, it can keep the obfuscation process transparent to the adversary.

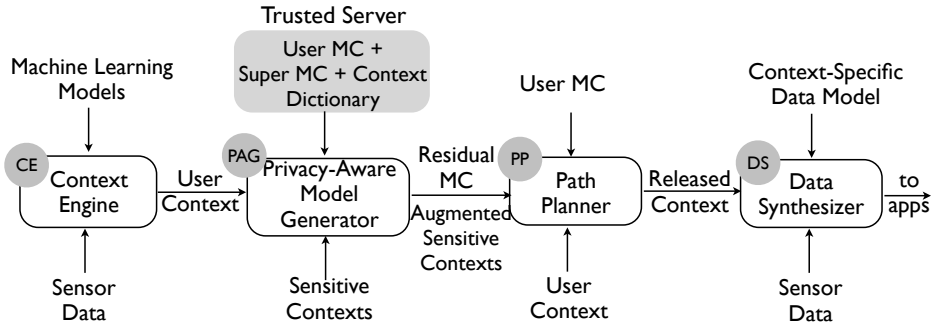


Figure 5.10: Extension of the iDeceit framework.

5.10 Extending to other sensors

We motivated the design of iDeceit for location privacy but the model-based falsification framework is a generic one and is applicable to data from other sensors as well. To generalize the falsification framework we need to extend some of the basic blocks in our framework (Figure 5.10). Below we describe the additional functionality and report evaluation results on accelerometer data.

5.10.1 Context Engine (CE)

To extend the notion of "user location", we utilize the more generic notion of "user context". A context is a label inferred from sensor data by combining together multiple sensor data streams. For example, GPS data together with maps are used to infer *location* contexts (e.g., home, work, hospital) [sag] which we have used in our work. Similarly, accelerometer and gyroscope data together is used to infer *activity* context (e.g., walking, running, still) [RMB10, BI04], acoustic data is used to infer *neighborhood* (inMeeting, withFriends) [LPL09], and *emotion* contexts (happy, sad, stressed) [CC11].

The Context Engine (CE) block transforms the data streams into contexts. The Markov chain model used by iDeceit is still feasible. However, each Markov chain state is now one of these generic contexts (e.g. walking, running, etc). The

formal definitions from Section 5.4 extends naturally to this case.

5.10.2 PAG and PP Blocks

The algorithms used in the DAG and the PP blocks are generic and can be used as it is by operating on the Markov chains where context states are used instead of location states.

5.10.3 Data Synthesizer (DS)

To extend the proposed framework to more complex contexts, the process of synthesizing data needs to be generalizing as well. In the specific case where each context state corresponds to a location, the data synthesis is simple as it is a one-to-one correspondence between GPS data and the context.

In the extended DS, we maintain a trained dynamical model for each context state. The model approximates the physical phenomenon that generated the original data and thus can be used to mimic the original data. Such a model can be obtained using the existing system identification techniques [Kat05]. Note the synthesis is more computationally intensive for sensors such as camera, microphone etc. and requires careful modeling.

5.10.3.1 Evaluating on activity dataset

To evaluate the usage of dynamical models as data generative models, we use our extended DS block on the activity dataset studied in [BI04]. The dataset is recorded for 20 different states (walking, running, standing, etc) using 6 sensors. For each state, we estimate 6 different state space models (one for each sensor) using subspace methods. We picked the model complexity to be equal to 30. Each model is then used to generate sensor data for specific sensor at certain state. A sample of the generated data from the model against the original sensor data is

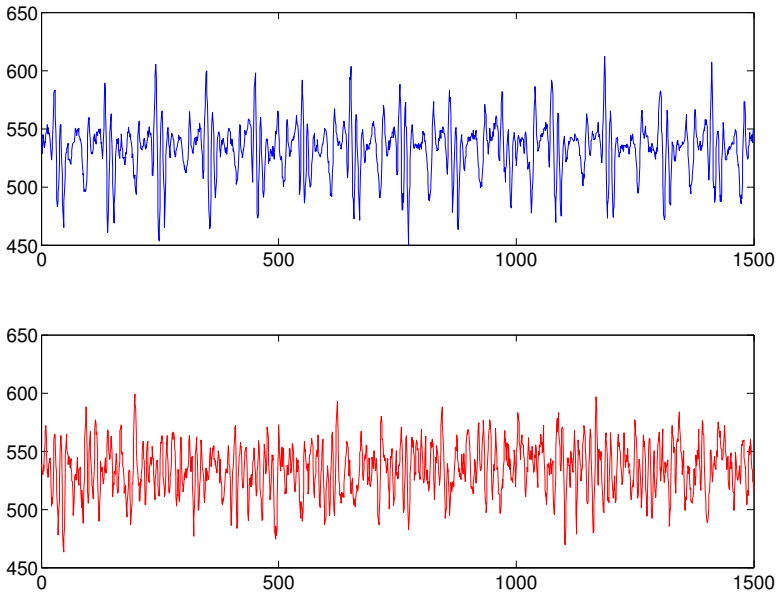


Figure 5.11: Actual accelerometer data from the activity dataset (top) versus a synthesized accelerometer data (bottom).

shown in Figure 5.11.

Utility of synthetic data: It is evident that the synthesized data in Figure 5.11 preserves the features of the original data like average value, data range and periodic spikes. However, the synthesized data is also noisy compared to the original signal. To test the utility of the generated data, we use the original sensor data to initially train a tree of SVM classifiers. This tree of SVM classifiers works as our Context Engine (CE) transforming data (over a time window) into one of the 20 context states. We then generate a random sequence of the states for which we want to synthesize data. We use the identified state space models to synthesize 1000 sensor samples for each state. Finally, we feed the CE with the both the synthesized data as well as the original data. If the CE output from the original data matches the output from the synthesized data, we can conclude that the model faithfully generates falsified sensor data that mimics a specific state. Our experiment showed a success rate of 97.38% using this activity dataset.

5.11 Discussion

In this chapter, we presented iDeceit, a framework that implements model-based data substitution to protect sensitive locations while ensuring the plausibility of the entire location trace. iDeceit uses a Markov chain to model the temporal transitions between locations and ensures that the falsified data conforms to the model. In fact, we formally show that model-based data release ensures zero-loss to privacy. We implemented a prototype on Android and performed our evaluation on data from a week-long user study. Our evaluation shows that iDeceit protects the privacy of sensitive locations and generates falsified paths that provide high utility and plausibility.

While iDeceit demonstrates the feasibility of using model-based substitution for location traces, the framework is also applicable to other sensor types. We show that by using a data generator iDeceit can be used with accelerometer data as well. In the future, we hope to explore richer models for capturing human behavior and better generative models for sophisticated sensors such as camera and microphones.

CHAPTER 6

Conclusion and Future Directions

The aim of this chapter is to reiterate the main theme and contributions of this thesis and layout directions that emerge as natural extensions to the work presented here. A summary of the contributions in each of the previous chapters is followed by an outline of future work.

6.1 Summary and Key Contributions

In this dissertation, we advocated a different perspective to the privacy problem. Instead of focussing on the data and its sensitivity we formulated the privacy problem in terms of the inferences that could be made using the shared data. In Chapter 2, we demonstrated that the users' privacy and utility preferences can be mapped into a blacklist and a whitelist of inferences respectively. The inference privacy problem then is to obfuscate data such that the whitelist is accurately inferable and the blacklist is protected.

In Chapter 3, we formulated the above problem using information theoretic notions. We simplified the setting by choosing the input from a finite and discrete space, explicitly specifying the joint distribution between the input and the inferences, and assuming that the input samples are independent and identically distributed. Under these conditions, we derived the feasible region, corresponding to the privacy and utility metrics, and also provided schemes to achieve the boundary points of the region. The solution has its own limitations. There are

scalability issues as the computational complexity increases exponentially with the size of the input domain. The joint distribution between the input and the inferences is often a generative model and the data samples are in practice rarely independent. Nevertheless, the solution provided us with an obfuscation technique (effectively synthesis) that would provide maximum utility under perfect privacy constraint under reasonable assumptions.

We then used the theoretical insights to design a system for providing privacy for time series data. The adoption of the theory was challenging. The input domain for even a few data samples from a sensor is large, and the samples are highly correlated in both time and space. So, any system looking to implement the results would need to work not on raw sensor data but some lower-dimension feature space. In addition, we needed a secure privacy-enforcement mechanism that could also provide monitoring of the resources used by an app, provide the users with a list of inferences that could be made using the shared data, have the ability to intercept raw data and apply privacy actions. In Chapter 4, we presented ipShield a privacy enforcing system on Android that provided the user with all the above options. To provide flexibility to the user we created a set of privacy actions that could be applied on the raw data itself. ipShied supported a simple binary recommendation engine, and thus relied on user interaction for rules creation.

In Chapter 5, we presented iDeceit, a framework that uses model-based privacy together with sensor data synthesis to handle the privacy challenges in sharing time series data. We model the relation between context states or inferences that can be made, using the sensor data, as a Markov chain. This ensures that the temporal correlation between the data is adequately captured. iDeceit then ensures plausible falsification of data by strictly following the transitions in the model which is representative of the user behavior. Whenever iDeceit releases a false state, it uses generative models to synthesize data, corresponding to the

false state, that is shared with the apps. To implement this on Android, the chapter discusses the design and implementation of an alternate datapath to push synthetic data to the apps.

This dissertation outlines the endeavour towards realization of privacy-enforcing system, rooted in strong theoretical foundations, intelligent enough to model human behavior, and over time minimizing user interaction while maximizing the ease of usability. In the course of this research, there are multiple natural extensions that have emerged. We mention some of them below.

6.2 Future Goals

- **Extension of the Theoretical Model:** Currently our theoretical model handles inference functions explicitly expressed as a table. Furthermore, the schemes provided for achieving the boundary points of the feasible region are exponential in the size of the table. We want to extend our formulation to use generative models of inference functions, and also provide scalable and computationally feasible (possible heuristic) privacy schemes for approximating the theoretical bounds. In addition, we want to incorporate side-channel information into the results. Often, privacy schemes such as differential privacy suffer from excessive loss of data utility while accounting for side-channel information. It will be interesting to devise schemes, even if they are for specific cases, which will be independent of auxiliary information but still provide utility.

- **Role of Trust and Game Theoretic Interpretations:** Mutual trust between parties involved plays an important role in any information flow. We encounter such scenarios often in military coalitions between countries, where sensitivity of the information together with trust determines how and with whom information will be shared. In such problem settings, often there exists a bi-directional information exchange between the provider and the consumer,

where the consumer performs computations on the provider data and shares the results with the provider. Trust is maintained by both parties and affects the quality of the information shared, which also acts as feedback for trust update. It will be interesting to explore if such a system can be modeled as a game and trust equilibria established under well-defined update policies.

- **Sensing Stack and Information Provenance:** Smartphones have evolved from mere communication devices into sensing platforms supporting a sprawling ecosystem of apps which thrive on the continuous and unobtrusive collection of personal sensory data. But what has not fully taken shape is a well-defined sensing stack, where starting from raw sensor data meaningful semantic abstractions, together with information provenance, are created and maintained respectively at each layer. We would like to enhance the ipShield [CSR14] architecture towards a sensing stack by integrating a feature extraction layer and an inference layer.

In the long term, push for affordable health care, automated and adaptive intervention and remote monitoring will see further expansion in mobile health apps and physiological data collection. For the military, need for automatic control, human-in-the-loop (e.g., secret agents) based data-to-decision models and ad-hoc coalitions will all mean data exchange between various, often untrusted sources. Smarter buildings will further instrument our personal spaces. As engineering solutions mature, newer forms of information will be collected from diverse domains needing better trust modeling, information provenance, and better uncertainty management. The contributions of this dissertation and the insights positions it as a unique knowledge repository to address the above challenges.

References

[AA04] D. Asonov and R. Agrawal. "Keyboard acoustic emanations." In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pp. 3–11, 2004.

[AH13] Yuvraj Agarwal and Malcolm Hall. "ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing." In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pp. 97–110, 2013.

[and] "Android 4.3's Hidden App Permission Manager." `http://www.androidpolice.com/2013/07/25/app-ops-android-4-3s-hidden-app-permission-manager-control\-permissions-for-individual-apps/`.

[aol] "AOL removes search data on vast group of web users." New York Times, Aug 8 2006.

[aos] "Android Open Source Project." `http://source.android.com/`.

[app] "Apple M7 Co-Processor." `http://en.wikipedia.org/wiki/Apple_M7`.

[ASB12] Adam J. Aviv, Benjamin Sapp, Matt Blaze, and Jonathan M. Smith. "Practicality of Accelerometer Side Channels on Smartphones." In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pp. 41–50, 2012.

[BBS09] Randy Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. "Persona: An Online Social Network with User-defined Privacy." In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pp. 135–146, 2009.

[BI04] Ling Bao and Stephen S Intille. "Activity recognition from user-annotated acceleration data." *Pervasive*, **LNCS 3001**:1–17, 2004.

[BKO10] David Barrera, H Güneş Kayacik, Paul C van Oorschot, and Anil Somayaji. "A methodology for empirical analysis of permission-based security models and its application to android." In *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 73–84. ACM, 2010.

[BRS11] Alastair R. Beresford, Andrew Rice, Nicholas Skehin, and Ripduman Sohan. "MockDroid: Trading Privacy for Application Functionality on Smartphones." In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, HotMobile '11, pp. 49–54, 2011.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. "Functional Encryption: Definitions and Challenges." In *Proceedings of the 8th Conference on Theory of Cryptography*, TCC'11, pp. 253–273, 2011.

[buf] "Under the Hood: Dalvik patch for Facebook for Android." `https://www.facebook.com/notes/facebook-engineering/under-the-hood-dalvik-patch-for-facebook-for-android/10151345597798920`.

[BW11] Niels Brouwers and Matthias Woehrle. "Detecting dwelling in urban environments using gps, wifi, and geolocation measurements." In *Proc. 2nd Intl Workshop on Sensing Applications on Mobile Phones*, pp. 1–5, 2011.

[CBS13] Supriyo Chakraborty, Nicolas Bitouzé, Mani Srivastava, and Lara Dolecek. "Protecting data against unwanted inferences." In *Information Theory Workshop (ITW), 2013 IEEE*, pp. 1–5, 2013.

[CC11] Drew Fisher Keng-hao Chang and John Canny. "Ammon: A speech analysis library for analyzing affect, stress, and mental health on mobile phones." *Proceedings of PhoneSense*, **2011**, 2011.

[CCS11] Supriyo Chakraborty, Haksoo Choi, and Mani B. Srivastava. "Demystifying privacy in sensory data: A QoI based approach." In *PerCom Workshops*, pp. 38–43, 2011.

[CKL11] David Chu, Aman Kansal, Jie Liu, and Feng Zhao. "Mobile apps: it's time to move up to CondOS." HotOS, 2011.

[CRJ13] Supriyo Chakraborty, Kasturi Rangan Raghavan, Matthew P. Johnson, and Mani B. Srivastava. "A framework for context-aware privacy of sensor data on mobile systems." In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, HotMobile '13, pp. 11:1–11:6, 2013.

[CRS12] Supriyo Chakraborty, Kasturi Raghavan, Mani Srivastava, Chatschik Bisdikian, and Lance Kaplan. "Balancing value and risk in information sharing through obfuscation." In *Information Fusion (FUSION), 2012 15th International Conference on*, pp. 1615–1622, July 2012.

[CSR14] Supriyo Chakraborty, Chenguang Shen, Kasturi Rangan Raghavan, Matt Millar, and Mani Srivastava. "ipShield: A Framework For Enforcing Context-Aware Privacy." In *Presented as part of the 11th USENIX Symposium on Networked Systems Design and Implementation*, 2014.

[dna] "Whered You Go With My DNA?" New York Times, Apr 24 2010.

[Dwo06] Cynthia Dwork. "Differential Privacy." In *ICALP*, pp. 1–12, 2006.

[EGC10] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. "TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones." In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pp. 1–6, 2010.

[EOM09] William Enck, Machigar Ongtang, and Patrick McDaniel. "On Lightweight Mobile Phone Application Certification." In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pp. 235–245, 2009.

[ES10] Deborah Estrin and Ida Sim. "Open mHealth Architecture: An Engine for Health Care Innovation." *Science*, **330**(6005):759–760, 2010.

[FCH11] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. "Android Permissions Demystified." In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pp. 627–638, 2011.

[FHE12] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. "Android Permissions: User Attention, Comprehension, and Behavior." In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pp. 3:1–3:14, 2012.

[GG03] Marco Gruteser and Dirk Grunwald. "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking." MobiSys '03, pp. 31–42, 2003.

[GH11] Craig Gentry and Shai Halevi. "Implementing Gentry's Fully-homomorphic Encryption Scheme." In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT'11, pp. 129–148, 2011.

[GJL12] U. Greveler, B. Justus, and D. Loehr. "Multimedia content identification through smart meter power usage profiles." In *Computers, Privacy and Data Protection*, 2012.

[GJP12] Saikat Guha, Mudit Jain, and Venkata N. Padmanabhan. "Koi: A Location-privacy Platform for Smartphone Apps." NSDI'12, 2012.

[GM82] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption &Amp; How to Play Mental Poker Keeping Secret All Partial Information." In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pp. 365–377, 1982.

[GMG13] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich. "Identifying personal genomes by surname inference." *Science*, **339**(6117):321–324, 2013.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems." *SIAM J. Comput.*, **18**(1):186–208, February 1989.

[GNG12] Michaela Götz, Suman Nath, and Johannes Gehrke. "MaskIt: Privately Releasing User Context Streams for Personalized Mobile Applications." In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pp. 289–300, 2012.

[GP09] Philippe Golle and Kurt Partridge. "On the Anonymity of Home/Work Location Pairs." In *Proceedings of the 7th International Conference on Pervasive Computing*, Pervasive '09, pp. 390–397, 2009.

[GST13] Daniel Genkin, Adi Shamir, and Eran Tromer. "RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis." Cryptology ePrint Archive, Report 2013/857, 2013.

[HHJ11] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, and David Wetherall. "These Aren'T the Droids You'Re Looking for: Retrofitting Android to Protect Data from Imperious Applications." In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pp. 639–652, 2011.

[HMN] Shashank Holavanalli, Don Manuel, Vishwas Nanjundaswamy, Brian Rosenberg, and Feng Shen. "Flow Permissions for Android." In *Tech Report*.

[HON12] Jun Han, E. Owusu, L.T. Nguyen, A. Perrig, and J. Zhang. "ACComplice: Location inference using accelerometers on smartphones." In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pp. 1–9, 2012.

[ipS] "ipShield: A Framework For Enforcing Context-Aware Privacy." `http://tinyurl.com/ipshieldgit`.

[JMV11] Jinseong Jeon, Kristopher K Micinski, Jeffrey A Vaughan, Nikhilesh Reddy, Yixin Zhu, Jeffrey S Foster, and Todd Millstein. "Dr. Android and Mr. Hide: Fine-grained security policies on unmodified Android." 2011.

[Kat05] T. Katayama. *Subspace Methods for System Identification*. Communications and Control Engineering. Springer, 2005.

[KB07] Boris Köpf and David Basin. "An Information-theoretic Model for Adaptive Side-channel Attacks." In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pp. 286–296, 2007.

[KKE10] Donnie H. Kim, Younghun Kim, Deborah Estrin, and Mani B. Srivastava. "SensLoc: Sensing Everyday Places and Paths Using Less Energy." In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pp. 43–56, 2010.

[KKM12] Krishnaram Kenthapadi, Aleksandra Korolova, Ilya Mironov, and Nina Mishra. "Privacy via the Johnson-Lindenstrauss Transform." *CoRR*, **abs/1204.2606**, 2012.

[KR13] John Krumm and Dany Rouhana. "Placer: semantic place labels from diary data." In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pp. 163–172. ACM, 2013.

[Kru09] John Krumm. "A survey of computational location privacy." *Personal Ubiquitous Comput.*, **13**(6):391–399, August 2009.

[KSS09] Younghun Kim, Thomas Schmid, Mani B. Srivastava, and Yan Wang. "Challenges in Resource Monitoring for Residential Spaces." In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '09, pp. 1–6, 2009.

[LFR12] Hong Lu, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast, Gokul T Chittaranjan, Andrew T Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. "StressSense: Detecting stress in unconstrained acoustic environments using smartphones." In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, Ubicomp '12, pp. 351–360, 2012.

[LJS12] Bin Liu, Yurong Jiang, Fei Sha, and Ramesh Govindan. "Cloud-enabled privacy-preserving collaborative learning for mobile sensing." In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, Sensys '12, pp. 57–70, 2012.

[LKR06] Kun Liu, Hillol Kargupta, and Jessica Ryan. "Random Projection-Based Multiplicative Data Perturbation for Privacy Preserving Distributed Data Mining." *IEEE Trans. on Knowl. & Data Eng.*, 2006.

[LL09] Tiancheng Li and Ninghui Li. "On the Tradeoff Between Privacy and Utility in Data Publishing." In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 517–526, 2009.

[LPL09] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. "SoundSense: Scalable Sound Sensing for People-centric Applications on Mobile Phones." In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, MobiSys '09, pp. 165–178, 2009.

[LWG13] Sangmin Lee, Edmund L. Wong, Deepak Goel, Mike Dahlin, and Vitaly Shmatikov. "πBox: A Platform for Privacy-preserving Apps." In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, NSDI'13, pp. 501–514, 2013.

[MBG13] Raul Montoliu, Jan Blom, and Daniel Gatica-Perez. "Discovering places of interest in everyday life from smartphone data." *Multimedia tools and applications*, **62**(1):179–207, 2013.

[MHM10] Min Mun, Shuai Hao, Nilesh Mishra, Katie Shilton, Jeff Burke, Deborah Estrin, Mark Hansen, and Ramesh Govindan. "Personal Data Vaults: A Locus of Control for Personal Data Streams." In *Proceedings of the 6th International COnference*, Co-NEXT '10, pp. 17:1–17:12, 2010.

[MKG07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. "L-diversity: Privacy beyond k-anonymity." *ACM Trans. Knowl. Discov. Data*, **1**(1), March 2007.

[MRS09] Min Mun, Sasank Reddy, Katie Shilton, Nathan Yau, Jeff Burke, Deborah Estrin, Mark Hansen, Eric Howard, Ruth West, and Péter Boda. "PEIR, the Personal Environmental Impact Report, As a Platform for Participatory Sensing Systems Research." MobiSys '09, pp. 55–68, 2009.

[MSF10] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. "Private Memoirs of a Smart Meter." In *Proceedings of the 2Nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pp. 61–66, 2010.

[MVB12] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. "Tapprints: Your Finger Taps Have Fingerprints." In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pp. 323–336, 2012.

[MVC11] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. "(Sp)iPhone: Decoding Vibrations from Nearby Keyboards Using Mobile Phone Accelerometers." In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pp. 551–562, 2011.

[NDA13] Shahriar Nirjon, Robert F. Dickerson, Philip Asare, Qiang Li, Dezhi Hong, John A. Stankovic, Pan Hu, Guobin Shen, and Xiaofan Jiang. "Auditeur: A Mobile-cloud Service Platform for Acoustic Event Detection on Smartphones." In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pp. 403–416, 2013.

[NKZ10] Mohammad Nauman, Sohail Khan, and Xinwen Zhang. "Apex: extending android permission model and enforcement with user-defined runtime constraints." In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pp. 328–332. ACM, 2010.

[NS08] Arvind Narayanan and Vitaly Shmatikov. "Robust De-anonymization of Large Sparse Datasets." In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pp. 111–125, 2008.

[NS10] Arvind Narayanan and Vitaly Shmatikov. "Myths and Fallacies of "Personally Identifiable Information"." *Commun. ACM*, **53**(6):24–26, 2010.

[Ohm09] Paul Ohm. "Broken Promises of Privacy: Responding to the surprising failure of anonymization." *Social Science Research Network Working Paper Series*, 2009.

[onx] "OnX." `https://www.onx.ms/`.

[pau] "Pausing Google Play: More Than 100,000 Android Apps May Pose Security Risks." `https://www.bit9.com/download/reports/Pausing-Google-Play-October2012.pdf`.

[PCG13] Abhinav Parate, Meng-Chieh Chiu, Deepak Ganesan, and Benjamin M. Marlin. "Leveraging Graphical Models to Improve Accuracy and Reduce Privacy Risks of Mobile Sensing." In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pp. 83–96, 2013.

[pdr] "PDroid patch for Android Jelly Bean." `http://github.com/gsbabil/PDroid-AOSP-JellyBean`.

[PPC12] Jun-geun Park, Ami Patel, Dorothy Curtis, Seth Teller, and Jonathan Ledlie. "Online pose classification and walking speed estimation using handheld devices." In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pp. 113–122, 2012.

[PRH11] K. Plarre, A. Raij, S.M. Hossain, A.A. Ali, M. Nakajima, M. Al'absi, E. Ertin, T. Kamarck, S. Kumar, M. Scott, Daniel Siewiorek,

A. Smailagic, and L.E. Wittmers. "Continuous inference of psychological stress from sensory measurements collected in the natural environment." In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pp. 97–108, 2011.

[pro] "Protocol Buffers." `https://developers.google.com/protocol-buffers/`.

[RAP11] Md. Mahbubur Rahman, Amin Ahsan Ali, Kurt Plarre, Mustafa al'Absi, Emre Ertin, and Santosh Kumar. "mConverse: Inferring Conversation Episodes from Respiratory Measurements Collected in the Field." In *Proceedings of the 2Nd Conference on Wireless Health*, WH '11, pp. 10:1–10:10, 2011.

[RGK11] Andrew Raij, Animikh Ghosh, Santosh Kumar, and Mani Srivastava. "Privacy Risks Emerging from the Adoption of Innocuous Wearable Sensors in the Mobile Environment." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pp. 11–20, 2011.

[RMB10] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, and Mani Hansen, Mark a nd Srivastava. "Using mobile phones to determine transportation modes." *ACM Trans. Sen. Netw.*, **6**(2):13:1–13:27, March 2010.

[RMM10] Kiran K Rachuri, Mirco Musolesi, Cecilia Mascolo, Peter J Rentfrow, Chris Longworth, and Andrius Aucinas. "EmotionSense: a mobile phones based adaptive platform for experimental social psychology research." In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pp. 281–290, 2010.

[RMX12] Ishtiaq Rouf, Hossen Mustafa, Miao Xu, Wenyuan Xu, Rob Miller, and Marco Gruteser. "Neighborhood Watch: Security and Privacy Analysis of Automatic Meter Reading Systems." In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pp. 462–473, 2012.

[run] "RunKeeper." `http://www.runkeeper.com/`.

[sag] "Saga Lifelogging." `http://www.getsaga.com/`.

[sec] "Android Security Overview." `http://source.android.com/devices/tech/security/`.

[SH12] Mudhakar Srivatsa and Mike Hicks. "Deanonymizing Mobility Traces: Using Social Network As a Side-channel." In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pp. 628–637, 2012.

[SM11] Rathindra Sarathy and Krishnamurty Muralidhar. "Evaluating Laplace Noise Addition to Satisfy Differential Privacy for Numeric Data." *Trans. Data Privacy*, **4**(1):1–17, 2011.

[SRP13] L. Sankar, S.R. Rajagopalan, and H.V. Poor. "Utility-Privacy Tradeoffs in Databases: An Information-Theoretic Approach." *Information Forensics and Security, IEEE Transactions on*, **8**(6):838–852, 2013.

[STL11] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. "Quantifying Location Privacy." SP '11, pp. 247–262, 2011.

[STT12] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. "Protecting Location Privacy: Optimal Strategy Against Localization Attacks." CCS '12, pp. 617–627, 2012.

[Swe02] Latanya Sweeney. "K-anonymity: A Model for Protecting Privacy." *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, **10**(5):557–570, October 2002.

[SWT01] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. "Timing Analysis of Keystrokes and Timing Attacks on SSH." In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, SSYM'01, pp. 25–25, 2001.

[tas] "Tasker." `http://tasker.dinglisch.net/`.

[TK] S Thurm and Y. Kane. "Your Apps Are Watching You." The Wall Street Journal, 2012.

[TSG09] Amin Tootoonchian, Stefan Saroiu, Yashar Ganjali, and Alec Wolman. "Lockr: Better Privacy for Social Networks." In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pp. 169–180, 2009.

[VP09] Martin Vuagnoux and Sylvain Pasini. "Compromising Electromagnetic Emanations of Wired and Wireless Keyboards." In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, pp. 1–16, 2009.

[Yao86] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets." In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pp. 162–167, 1986.

[YCG12] Tingxin Yan, David Chu, Deepak Ganesan, Aman Kansal, and Jie Liu. "Fast app launching for mobile devices using predictive user context." MobiSys '12, pp. 113–126, 2012.

[ZCC12] Andong Zhan, Marcus Chang, Yin Chen, and Andreas Terzis. "Accurate Caloric Expenditure of Bicyclists Using Cellphones." In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pp. 71–84, 2012.

[ZZJ11] Yajin Zhou, Xinwen Zhang, Xuxian Jiang, and Vincent W. Freeh. "Taming Information-stealing Smartphone Applications (on Android)." In *Proceedings of the 4th International Conference on Trust and Trustworthy Computing*, TRUST'11, pp. 93–107, 2011.