UNIVERSITY OF CALIFORNIA, SAN DIEGO

Policy-Aware Sender Anonymity in Location-based Services

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

in

Computer Science

by

Avinash Vyas

Committee in charge:

2011

DEDICATION

To my parents, Mrs. Roop Kaur Vyas and Mr. Suresh Vyas, who have endured many hardships to help me reach here. To my Wife, Minakshi Bohra, for her valuable support. To little ones, Anshu and Aarush, for allowing me the time to do this work.

EPIGRAPH

Some goals are so worthy, it's glorious even to fail.
—Lieutenant Manoj Kumar Pandey,
Indian Army.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

During my PhD years, my advisor, mentors and co-workers helped me to define and reach work goals, develop my research skills and shape my career. I am sincerely grateful to all of them for contributing to my success.

My earnest acknowledgement goes to my advisor, Alin Deutsch, and my mentor Richard Hull, who from the beginning guided me in finding a dissertation topic, forming a doctoral committee and carry out the thesis proposal. Their encouragement and suggestions helped me a lot in learning the right way of tackling difficulties one faces in research environment. Research meetings and long discussions with them, were one of the most instructive part of my graduate education.

I would like to thank my my senior colleagues at Bell Laboratories, Lucent Technologies, specially Michael Benedikt, Jerome Simeone, Juliana Friere and Mary Fernandez (at ATT Research) for giving me (prior to coming to UCSD) the opportunities to participate in very novel research projects. I would like to specially thank Michael Benedikt, who along with Richard Hull, was instrumental in persuading the management at Bell Laboratories to support me for the PhD program at UCSD. And special thanks to my manager Lalita Jagadeesan for continuing the support and providing constant encouragement and guidance at Bell Laboratories, Alcatel-Lucent.

I would like to thank Kevin Kaliang Zhao, my colleague in Database Group at UCSD, for his insights and contributions towards this study. I would also like to thank my other colleagues in CSE Dept at UCSD, Mayank Kabra, Nicola Onose, Kian Win, Yannis Katsis, Heasoo Hwang, Emiran Curtmola, Nathan Bales, Elio Demmagio and Liang Chen for the friendly banter, discussions, gossips and their invaluable help at various moments in my stay at UCSD.

Last, but not the least, I would like to thank all the members of my doctoral committee, specially Yannis Papakonstaniou, for their advice, suggestions and questions during the thesis proposal phase and the defense talk.

Chapter 3, in full, is a reprint of the material as it appears in IEEE International Conference on Data Engineering 2010. Deutsch, Alin; Hull, Richard; Vyas, Avinash; Zhao, Kevin Keliang. The dessertation author was the primary investigator and author of this paper.

Chapter 4, in part is currently being prepared for submission for publication of the material. Kevin Kaliang Zhao at UCSD and Richard Hull at IBM, together with Alin Deutsch, made an important contribution to the material. The dessertation author was the primary investigator and author of this material.

VITA

1999	B. E. in Computer Science and Engineering, M.B.M. Engineering College, Jodhpur, Rajasthan, India.
2001	M. Tech. in Computer Science and Engineering, Indian Institute of Technology, Kanpur, Uttar Pradesh, India.
1999-2001	Teaching and Lab Assistant, Indian Institute of Technlogoy, Kanpur, Uttar Pradesh, India.
2001-2011	Member of Technical Staff - 1, Bell Laboratories, Murray Hill, New Jersey, United States of America.
2011	Ph. D. in Computer Science, University of California, San Diego, California, United States of America.

PUBLICATIONS

Richard Hull, Bharat Kumar, Daniel Lieuwen, Peter Patel-Schneider, Arnaud Sahuguet, Sriram Varadarajan, Avinash Vyas, “Rule-Based Service Customization via Houdini”, *WWW Poster*, 2003.

Richard Hull, Bharat Kumar, Daniel Lieuwen, Peter Patel-Schneider, Arnaud Sahuguet, Sriram Varadarajan, Avinash Vyas, “Enabling Context-Aware and Privacy-Conscious User Data Sharing”, *IEEE Conference on Mobile Data Management*, 2004.

Mary Fernandez, Jerome Simeon, Avinash Vyas, “The Simplest XML Storage Manager Ever”, *XIME-P*, 2004.

Michael Benedikt, Angela Bonifati, Sergio Flesca, Avinash Vyas, “Adding Updates to XQuery: Semantics, Optimization and Static Analysis”, *XIME-P* 2005.

Michael Benedikt, Angela Bonifati, Sergio Flesca, Avinash Vyas, “Verification of Tree Updates for Optimization”, *International Conference on Computer Aided Verification*, 2005.

Alin Deutsch, Richard Hull, Avinash Vyas, Kevin Keliang Zhao, “Policy-Aware Sender Anonymity in Location Based Services”, *IEEE International Conference on Data Engineering*, 2010.

ABSTRACT OF THE DISSERTATION

Policy-Aware Sender Anonymity in Location-based Services

by

Avinash Vyas

Doctor of Philosophy in Computer Science

University of California, San Diego, 2011

Alin Deutsch, Chair

Sender anonymity in Location-based services (LBS) refers to hiding the identity of a mobile device user who sends requests to the LBS provider for services in her proximity (e.g. “find the nearest gas station etc.”). The goal is to keep the requester’s interest private even from attackers who (via hacking or subpoenas) gain access to the LBS request and to the locations of the mobile user and other nearby users at the time of request. In an LBS context, the best-studied privacy guarantee is known as *sender k-anonymity*. We show that the state-of-the art solutions for sender k-anonymity defend only against the naive attackers who have no knowledge of the anonymization policy that is in use. We strengthen the privacy guarantee to defend against more realistic “policy-aware’ attackers.’. We describe a polynomial algorithm to obtain an optimum anonymization policy. Our implementation and experiments show that the policy-aware sender k-anonymity has the potential for practical impact, being efficiently enforceable, with limited reduction in utility when compared to policy-unaware guarantees.

Next we extend the policy-aware privacy guarantee to a class of attackers who knows historical user locations i.e. trajectory-aware. We call it the trajectory-aware policy-aware sender k-anonymity guarantee. We describe how this novel privacy guarantee is provided in an *offline* scenario, when a LBS provider logs the LBS requests sent by its users over a period of time and later wants to publish/share these logs. While these logs can be extremely valuable for advertising, data mining research and network management, we show they pose serious threat to anonymity of the LBS users. We describe a method to anonymize these LBS request logs and preserve trajectory-aware policy-aware sender k-anonymity in the offline scenario. We show that finding the optimum offline policy that provides trajectory-aware policy-aware k-anonymity is *NP-Hard*. Hence, we describe a novel bounded approximation algorithm and empirically show the effectiveness of this approximation algorithm for anonymizing large sizes of data (up to 1 million users).

Chapter 1

Introduction

Recent years have witnessed increased demand for Location-Based Services. Intuitively, a Location-Based Service refers to a information service that answers requests of mobile device users for services, facility, or other users in their proximity (e.g. “find the nearest gas station” or “alert me when a friend is within 1 mile of my location”’). Formally, it is defined in [lbs] as follows.

1.1 Location-based Services

Location-based services (LBS) refers to an information or entertainment service, accessible with mobile devices through the mobile network and utilizing the geographic location of the mobile device . The location is used by the LBS in one of the following possible ways:

- sort or filter the result of a user query to those that are closest to user location.
- show/announce the location of a user as a pointer on the map to other users.
- trigger an event when a user moves inside/outside a predefined region.

The above definition describes a LBS in its most common forms, focusing on the 1st generation of LBS that are primarily available through the mobile networks. But the field of LBS is still evolving and therefore some of the newer applications (e.g. location-enabled web services using HTML 5) may not be described using above definition. Next we describe some of the popular classes of commercial LBS that uses the location data in one of the above described ways.

Proximity Search. The location-based services in this class are examples of LBS that sort or filter the result of user queries, using their location. The most common example in this category is an application that helps users to find services they seek in their proximity e.g. "nearest gas station", or "Thai restaurant". The location is not an enabler for this service but rather acts as a filter for the result of the user queries. Google Maps [gma] is an example of such application. Groupon [gro] is another popular application that allows users to find discount coupons for the shops in their proximity.

Social Networking. The location-based services in this class are examples of LBS that allows a user to share/announce their location to other users. These applications use the location to enable and/or enhance the social networking scenario. It is widely believed that the location provides added value to the initiation of a social interaction. For e.g. a person is more likely to meet/interact with friends in close proximity to him. Google Latitude [gla], Facebook Places [fb], Gowalla [gow] and FourSquare [fou] are examples of such LBS. All these applications allow a user to disclose his/her location on a map to (a subset of) other users, who then can use this information in a manner deemed fit to them.

Marketing and Advertising. The location-based services in this class are examples of LBS that trigger an event when a user moves inside/outside a predefined region. For e.g. when some users pass by a store, they are sent a discount coupon. The location-based marketing and advertising has emerged as the latest tool in support of the traditional marketing and advertising strategies. Applications like Shopkick [sho] and Foursquare [fs] track a user's visits to retail shops and businesses and encourage them to visit new retail shops and their favorite shops frequently by providing them incentives such as discount, freebies and access to special events.

Most of these LBS applications uses the query-response model i.e. the user (or a client application) sends a request to an LBS server that contains details of the requested information or operation. In the response the LBS sends back the status of the operation or the requested information back to the client. Some of the LBS applications use subscriptions, allowing a user to subscribe for certain event(s). The server later sends response(s)/notification(s) when the event(s) occurs. But as the users move, they need to send location updates to the server, thus as such these LBS can also be modeled as query-response. We elaborate more on the working a LBS in the next section.

1.2 Prevailing Model of LBS

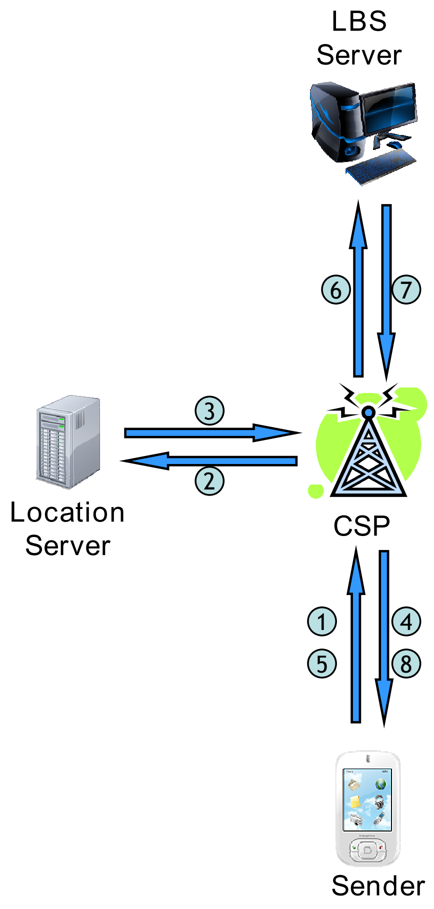


Figure 1.1: LBS Model

This section introduces the prevailing model of a LBS, describing the entities involved in the delivery of the service. Even though a LBS does not dictate how the location of the user is obtained (e.g. it can be provided explicitly by the user), in this model we assume that the location is computed automatically using the mobile device and/or the wireless network. Infact improved location computing technologies for mobile devices is a major contributor to the popularity of LBS.

As shown in Figure 1.1 there are four core elements in the delivery of a location-based service: the user making a request, typically called the *sender*, the (wireless) Communication Service Provider, denoted as *CSP*, the Location Server, denoted as *LS*, and the Location Based Service (LBS) provider, denoted as *LBS*. Next we give a functional overview of these core elements and describe the the message flow and computations behind an interaction of a user and a LBS.

1.2.1 Sender

The sender initiates the LBS request, typically using an application (client part of the LBS service) from his mobile device. These client applications can be native to a mobile platform (e.g. iOS or Android) or they can be implemented as web applications (e.g. HTML5) running in the browser. Whether the LBS request is sent using a native application or a web application, it typically contains a unique device identifier (e.g. IP-address, MAC-address, IMEI number (International Mobile Equipment Identity)). Unlike traditional wireline telephones, the mobile devices are not shared, hence a unique identifier associated with a mobile device represents a unique user. These applications obtain the "current" location of the mobile device from the run-time environment on the mobile device, which in turn obtains it from the Location Server.

1.2.2 Location Server

The Location Server (LS) refers both to the infrastructure that computes the location of a mobile device and to the interface that provides access the computed location. There are two classes of location computing infrastructure, *Network-based* and *Device-based*. Next we describe the two most prevalent methods for computing mobile device location, one from each of the above mentioned class.

Network based. The wireless network based location computing infrastructure was developed to compute the location of mobile device locations for E911 [E91] purposes. There's more than one technology to develop such infrastructure, we briefly describe on such technology known as *Time Difference of Arrival* (TDOA). The wireless networks that uses TDOA deploys specialized network components known as Location Measuring Unit (LMU) and Position Determining Entity (PDE). These components computes the location of a mobile device by measuring the time of arrival of a radio signal from a mobile device at three or more separate cell towers. Using the (known) speed of propagation of radio waves, the difference in the arrival times at a pair of cell towers allows one to draw a curve on which the transmitting device is located. The intersection of 2 or more of these curves gives the location of the device. The computed location can be accessed through specialized network component such as the *Mobile Positioning Center* (MPC) in the CDMA network.

Handset based. Handset based solutions are commonly used in smartphone platform (e.g. iOS and Android) to provide location data to the smartphone applications. This approach uses the mobile device to measures the strength of various radio signals (WiFi, 2G, 3G etc) from the nearby WiFi access points and cell towers and sends them to a server, along with the identities of the access points. The server has a database of locations of WiFi access points and cell towers and it uses the the data sent by the mobile device to compute the location of the mobile device relative to the location of the access points. The computed location is then sent back to the mobile device. *SkyHook* [sky] and *Google Location Service* [gls] compute location of a mobile device using this method. The location computed with this method is primarily used in non-emergency LBS, because it is not guaranteed to be accurate (e.g. if a WiFi access point's is moved and its location in server's database is not updated then its going to lead to wrong result).

1.2.3 LBS Provider

The LBS provider implements the required functionality to provide the required service. It uses the location in the LBS request to respond to the user query or perform the requested operation. In addition, some of the LBS providers logs the LBS requests and use them to enable new services or improve existing services. This is a common practice as indicated in the prevailing data retention and privacy policies of the popular LBSs such as *Facebook Places*, *FourSquare*, *Loopt*, *Google Maps* and *Gowalla*.

These core elements, together form the LBS ecosystem. The message flow that we describe next represents how these elements interact when a sender initiates a LBS request.

- The sender invokes the client application on the mobile device to send a LBS request. But before the LBS request can be sent to the LBS provider, it needs to obtain the current location of the mobile device. The run-time environment on the mobile device obtains this location from the Location Server, on behalf of the client application. This is shown as messages (1) and (2) in Figure 1.1.

- The Location Server (LS) which is either the MPC in the CSP's network or an Over-The-Top (OTT) service such as *SkyHook* or *Google Location Service* computes the location of the mobile device and sends it back to the user device. This is shown as messages (3) and (4) in Figure 1.1.

- The client sends the obtained location in the LBS request (e.g. "nearest Chinese restaurant") to the LBS provider through the CSP's network. This is shown as messages (5) and (6) in Figure 1.1.

- The LBS provider compute the response of the LBS service request using the location sent with the request. The response is then sent back to the sender over the CSP's network.

1.3 Privacy Incidents

While users have liked the innovative uses of location data in the form of location-based services, there is an increasing concern that these services brings new threats to user privacy. And to show that these concerns are not entirely theoretical, we list some recent incidents reported in the media that highlights some of the key privacy concerns.

1.3.1 Crimes: Robbery, Heist, Stalking

Several incidents of robbery, heist and stalking have been attributed to the disclosure of location on social networking sites such as Facebook, Twitter and FourSquare. These incidents were a result of victims revealing their location on the social networking sites without realizing that every user on the site had the permission to see the post. The burglars used these posts to identify the victims that were not at their homes. This is highlighted on *Please Rob Me!* [rob] website that has numerous examples of users inadvertently disclosing their location on social networking sites. The use of social networking site for crimes is in fact such a growing trend that insurance companies, such as *More Than* [mth], are thinking about taking this factor into account in the insurance costs, after their survey revealed that monitoring social networking sites is now a big business for criminals.

1.3.2 Commercial Big brother

In addition to these ignorant disclosure of LBS requests by the users, several independent research studies [loc, wsj, EGC+10] have revealed that many LBS providers are logging user LBS requests along with some form of device identifier (IP-address, MAC-address). This includes companies such as *Google Maps*[gma], *Foursquare* [fs] and *Facebook Places* [fb]. Not only the LBS providers, even the companies such as *Google* and *Apple* that provide application platforms for mobile devices are tracking user location every time a user sends an LBS request. And finally, as reported in a study [wsj] in Wall Street Journal, the LBS client applications on the mobile device are collaborating with the advertisers and are sending the LBS request data to them. Over a period of time, the locations visited by the user and the LBS requests can reveal a lot about the user, including his social and political interests and other information such as his employer and medical condition (to some extent).

1.3.3 Government Big brothers

The LBS providers and advertisers are not alone in seeking the LBS data of the users. Law enforcement agencies have also started aggressively seeking LBS data from the LBS providers and the Location Servers. The ACLU report [acl] summarizes a partial list of incidents (reported in media) when authorities accessed or tried to access such information. For e.g. in 2008, FBI sought and received location-tracking information not just for a robbery suspect, but for 180 other innocent people without a warrant. Such practices are most likely kept secret to avoid disclosing the investigatory techniques of the Law enforcement agencies and avoid potential backlash from consumers. Therefore, one can assume that if needed, these agencies can approach the LBS providers to gain access to user LBS requests.

1.4 User Privacy in Location-based Services

All these incidents have drawn the attention of the users, privacy advocates, researchers and the legislative bodies towards user privacy issues. The users and the privacy advocates have been demanding better privacy measures, while researchers have been working on finding solutions to these privacy problems. The two problems that have attracted the maximum attention from researchers are *Sender Anonymity* and *Location Privacy*.

1.4.1 Sender Anonymity

Consider a LBS service that helps users to find services they seek in their proximity (e.g. “nearest Thai restaurant” or “nearest Pet Store”). The requests sent to this LBS by the users can reveal their personal interests such as “what food they like”, “whether they have a pet”, to the LBS. Some of these interests such as political affiliation may be considered sensitive by the users and therefore may not want the LBS (or any other person) to know about it. Thus while they want to access the LBS i.e. send LBS requests, they do not want the LBS (or any other person) to know that the request came from them. We refer to this problem of hiding the identity of the sender of a LBS request as sender anonymity.

The important aspect of any sender anonymity solution is the attackers against whom the sender anonymity is achieved. One solution may provide sender anonymity against an eavesdropper who can intercept the LBS requests but cannot view its content. But the same solution may fail to hide the identity of a sender from the LBS providers who can view the content of the LBS. Or it may fail to provide sender anonymity against a law enforcement agency that can use subpoena to obtain the LBS requests from the LBS provider and user locations from the Location server.

1.4.2 Location Privacy

Consider a LBS service that allows users to disclose their location on a map to (a subset of) their family and friends (e.g. “I am at UTC mall” or “I am at Qualcomm stadium”). The users may not want to disclose their location all the time or to every body (e.g. a user may want to tell his friends that he is at Qualcomm stadium but does not want his spouse to know that). We refer to this problem of hiding the location of a mobile device (of a user) from an attacker as location privacy. Note that in this problem the focus is on preventing the disclosure of a user location in the LBS ecosystem. It does not prevent location disclosure against other threats such as physical surveillance.

The two problems of location privacy and sender anonymity described above are complementary to each other. In location privacy the goal is to hide the precise location of the user but one is not required to hide the identity of the user, while in sender anonymity the goal is to hide the identity of the user but one is not required to hide the location.

1.4.3 Scope

In this study, we focus on the problem of sender anonymity in LBS. But since the problem of sender anonymity has been studied extensively in traditional messaging systems such as E-mail and Web-transactions, we first briefly describe the the characteristics of these solutions and then show why they cannot provide sender anonymity in LBS.

Chapter 2

Anonymous Communication

In the context of e-communication, the early cryptographic encryption systems for public networks were focused on hiding the content of a message but not on hiding who is communicating with whom and how often. In particular, without looking at the content, the sender of a message can be identified, and the sender and receiver can each be identified as participating in some communication. The later problem is referred to as *Sender Receiver Unlinkability* and the former problem as *Sender Anonymity* [PW87].

Next we look at the some of most popular and publicly available solutions [Cha81, RR98, GRS99, DMS04] for sender anonymity in the the context of messaging systems such as E-mail and World Wide Web. Some of these solutions [GRS99, DMS04] are independent of applications and are applicable to any messaging system. In describing these solutions we focus particularly on the attacker against whom the sender anonymity is achieved. A solution may provide sender anonymity against a local eavesdropper that can observe the messages received by a receiver but not against a global observer that can observe all the messages in the system.

2.1 Anonymity in messaging systems

2.1.1 Mix Network

First proposed in the context of anonymous email-system [Cha81], a mix network consists of a network of special purpose routers called *mix*. A *mix* (shown in

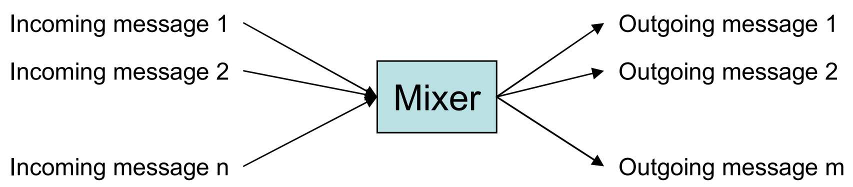


Figure 2.1: Mix Router

Figure 2.1 is a specialized router that uses cryptographic transformations and reordering to make it difficult to relate an outgoing message with an incoming message. Thus a mix acts a black box and does not allow an observer to track a particular message using specific bit-pattern, size or ordering with respect to other messages. A message is passed through a series of Mix routers making it even more difficult to track a particular message in the network.

The Mix network provides sender receiver unlinkability against a global observer since it is not possible to identify who is communicating with whom. It also provide sender anonymity against the receiver since the receiver cannot trace the sender of a message. The working of mix network requires that the mix are trusted and are never compromised and are available all the time.

2.1.2 Crowds

Crowds is a system to provide sender anonymity to the users for web transactions. The approach is based on the idea of “blending in the crowd” i.e. hiding one’s actions within actions of many others. To execute anonymous web transactions, a user first joins a set of other users known as the *crowd*. A user’s request to the web server is first passed to a random member of the crowd. That member can either *submit* the request directly to the end server or *forward* it to another randomly chosen member, and in the latter case the next member chooses to submit or forward independently. When the request is submitted, it is submitted by a random member. Figure 2.2 shows schematic representation of this process.

The web server receiving a request is unable to identify the true sender because it is equally likely to have originated from any member of the crowd. Even the collaborating crowd members cannot distinguish the originator of a request from a member who

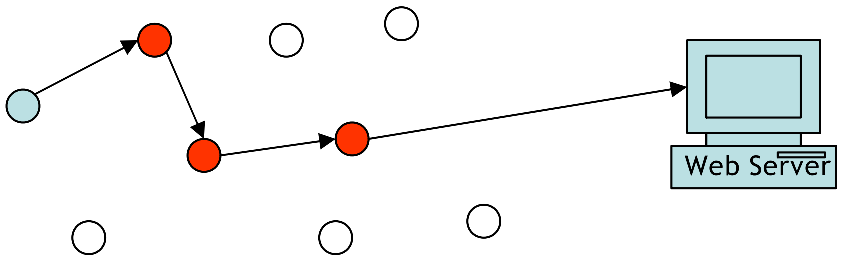


Figure 2.2: Crowd System

is simply forwarding the request. Thus crowds provide sender anonymity against the web server (even in the presence of collaborating members) i.e. the receiver. It neither provides sender anonymity nor sender receiver unlinkability against a global observer.

2.1.3 Onion Routing

Onion routing is a general purpose infrastructure for anonymous communication over a public network such as Internet. It consists of a network of special purpose routers called *onion routers* that are connected by a longstanding (permanent) connections. To provide anonymity the messages sent through the network are multiplexed over these longstanding connections.

Onion routing's anonymous communications are protocol independent and exist in three phases: connection setup, data movement and connection tear-down. The setup consisting of choosing the sequence of onion routers in between the sender and the receiver. All the communication passes through this route established by the setup phase. The route is setup in a way that each onion router can only identify the previous and the next hops along a route. To every message, the sender adds a layer of encryption for each onion router on the route. As the message moves through the predetermined path, each onion router removes one layer of encryption, so it arrives at the receiver as plain-text.

Since the receiver is only aware of the the last onion router on the path, this approach provides sender anonymity against the receiver. Due to layering cryptographic operations, when the data moves through the network it appears different to each onion router. Therefore, it provides sender anonymity and sender receiver unlinkability against eavesdropper and global observers.

2.1.4 Anonymity measure

The anonymity measure used in these studies for sender anonymity is known as *anonymity set* [Cha88]. In the context of a messaging system, the anonymity set is defined as the set of users who could have possibly sent a particular message. The size of an anonymity set is used as an indicator of the level of anonymity provided to the sender. In the worst case, the size of the anonymity set is 1, which means no anonymity and in the best case it equal to the number of users in the system, which means any user could have sent the message. The goal of these anonymity solutions is to increase the size of the anonymity set.

2.2 Sender Anonymity in LBS

The LBS model described earlier is also an example of a messaging system, therefore one can try to use one of the above described solutions to provide sender anonymity to LBS requests. But these systems provide sender anonymity against attackers who has access to the LBS requests but cannot access its content. They fail to provide sender anonymity when the attacker has access to the content of the messages (e.g. receiver) as the contents may reveal the identity of the sender (e.g. if the message includes userid or other forms of user identifier). This is exactly the case with a LBS provider who can access the contents of the LBS requests.

It was first shown in [GG03] that the location of a sender included in a LBS requests can also reveal the identity of the sender in some cases. Consider a scenario when a user sends a LBS request from his home at 11:00 pm in the night. The location in this LBS request can be correlated with a database of geo-coded postal address to identify the residence. This address can be further used to do a lookup in property listings, to reveal the owner who most likely is the sender. Thus, if a user sends a LBS request from a location that exclusively belongs to him (at the time he sends the request), his anonymity can be breached using the location in the LBS request. Therefore the solutions that provide sender anonymity in a messaging system, alone cannot provide sender anonymity to the users of LBS.

Thus research in the area of sender anonymity for LBS focuses on preventing an attacker from using the location of the sender to breach sender anonymity. The most studied approach to address this problem is known as *Spatial Cloaking*.

2.2.1 Sender anonymity via Spatial Cloaking

Spatial cloaking [GG03] refers replacing the location in a LBS request with a region (of some predetermined shape) that includes the location of the sender. Thus a attacker who has access to the LBS request cannot use the region in the request to determine the sender. But this approach may fail if the sender is the only user in the region and the attacker can observe the locations of the users within that region.

Thus to prevent such a case, the region that is used in the LBS request is picked such that it includes k-1 users in addition to the sender. We refer to this as k-anonymous spatial cloaking. When an attacker who has access to such a LBS request and locations of all the users in that region, tries to reverse engineer the possible sender, there are k users (any one of) who could have sent the request. Without any additional information, the attacker cannot narrow down his choices of possible senders to less than k. Thus it provides sender anonymity by hiding the sender in a crowd of k potential senders. We refer to this as *sender k-anonymity*.

Example 1. Figure 2.3 shows an example of spatial cloaking by an anonymizer, that replaces the location **A** in the LBS request with a region containing the location **A** and location of 7 other nearby users.

Figure 2.4 shows the architecture of a privacy conscious LBS that uses spatial cloaking based anonymizer. The anonymizer acts as the proxy between the users and the LBS server. When a user wants to access a LBS, he sends the LBS request with his precise location to the anonymizer. The anonymizer computes a region that contains the locations of the sender and k-1 additional users. The anonymizer replaces the exact location of the user in the LBS request with this region. The LBS request containing region is forwarded to the LBS server.

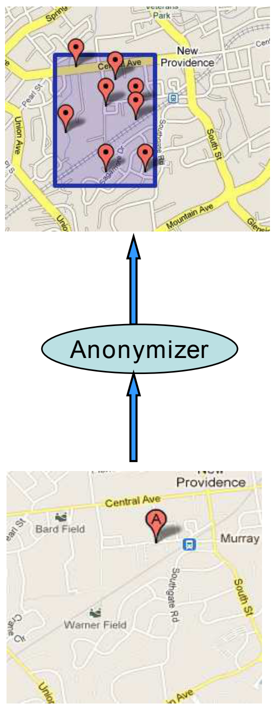


Figure 2.3: 8-anonymity Spatial cloaking

2.2.2 Trade-off: Anonymity vs Cost

If privacy was the only consideration, one could replace the location in the LBS request with a very large region that contains hundreds of additional users (e.g. the entire neighborhood around the user). The use of a region in the LBS request affects the quality of the response (i.e. utility) and/or the cost of providing the LBS service. For e.g. a LBS query requesting the locations of nearby coffee shops that uses a bigger region will yield far more results than a similar query that uses a location. The greater number of results requires more processing at the server, increased transmission cost of the result and decreased value to a user. Thus another consideration, when selecting a region to replace a location in a LBS request, is to try and minimize the area of the region. But as described in the next example, a spatial cloaking that replaces locations in the LBS requests with the smallest region containing the user location and those of $k-1$ other users may not provide sender k-anonymity as intended.

Example 2. Consider the spatial cloaking for locations shown in Figure 2.5 that provides sender 2-anonymity by picking the smallest region containing the sender and one

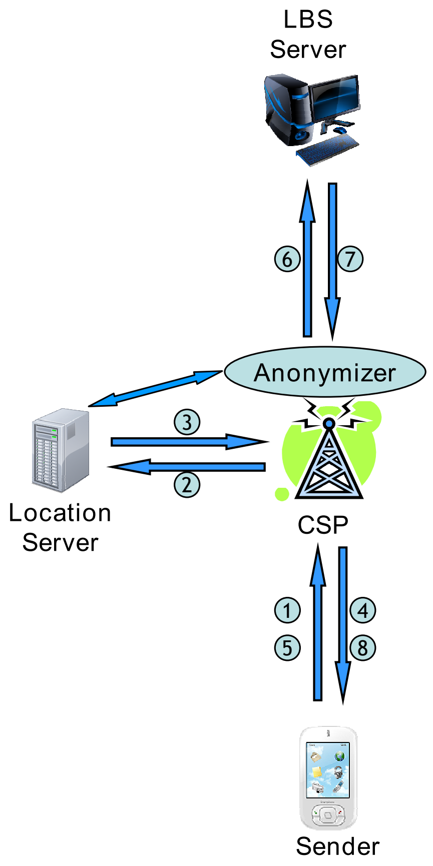


Figure 2.4: Spatial cloaking based anonymizer

additional user. For the LBS requests sent from location **A** and **B**, the anonymizer uses the region R, and for LBS request sent from location **C** it uses the region S. If an attacker observes an LBS request with region S, he can successfully reveal that the sender is **C**.

This brings us to an interesting conundrum. While a bigger region can provide better sender k-anonymity in comparison to a small region, it increases the processing cost of a LBS request and can reduce its value to its users. On the other hand, while smaller regions reduces the LBS processing cost, trying to reduce the size of the regions too aggressively, could result in breach of sender k-anonymity. These opposing considerations results in a trade-off between privacy and cost/utility, as we want to minimize the area of the region while preserving sender k-anonymity.

Thus a spatial cloaking based solution should not only provide sender k-

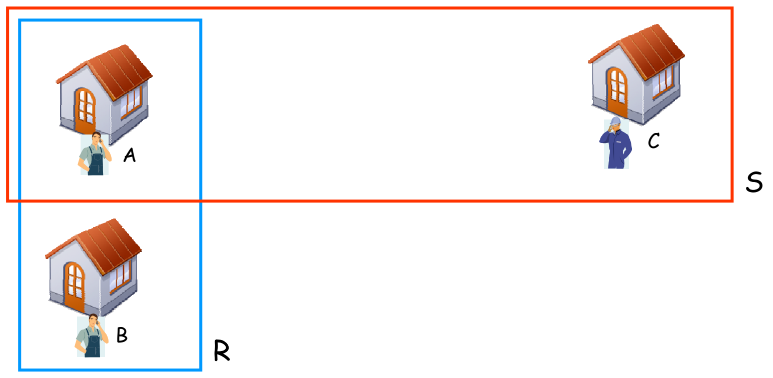


Figure 2.5: Anonymity breach due to agressive minimization

anonymity, but should also ensure that the regions picked for spatial cloaking are optimal. As we describe next, this is one of the key challenges that we address in our study.

2.2.3 Problem statement

As mentioned earlier, a key aspect of sender anonymity solution is the type of attacker against whom it is achieved. In this study we consider two classes of attackers that represents current (realistic) threats to sender anonymity of the users of LBS.

- *Policy-aware attacker*: an attacker who has access the LBS request, the locations of all the users at a given instant of time and who knows the complete design of the system used to provide this protection.
- *Trajectory-aware and Policy-aware attacker*: a policy-aware attacker who has access to the past LBS requests sent by a user during a given interval of time and the locations visited by all the users during that interval.

Snapshot Policy-aware Sender k-anonymity

We refer to the sender k-anonymity guarantee against the class of policy-aware attackers as *policy-aware* sender k-anonymity. Ensuring sender anonymity against policy-aware attacker is non-trivial since such an attacker is aware of each and every aspect of defense mechanism. The privacy breach in Example 2 is also due to the attacker being aware of the spatial cloaking function (i.e. region picked for each location). The key challenges addressed in this study are :

[1] we define the notion of policy-aware sender k-anonymity and propose a spatial cloaking based solution that preserves policy-aware sender k-anonymity. We aim for a strong information theoretic privacy guarantee that depends upon the information available to an attacker but there is no limiting assumption on the available processing power.

[2] Since spatial cloaking reduces utility of the LBS response and/or increases the cost of processing a LBS request, we propose a spatial cloaking solution that finds the optimum spatial cloaking for the LBS requests without

[3] study the affect of the choice of regions (e.g. circular, quadrants) on the complexity of finding the optimum spatial cloaking.

This version of policy-aware sender k-anonymity only applies to requests sent by the users at a given instant of time. It does not consider the LBS requests sent by the users in the past (other instant of time). Therefore we extend the policy-aware sender k-anonymity against an attacker who also has access to the history of location visited by the users (trajectory) and their prior requests.

Trajectory and Policy aware Sender k-anonymity

As indicated by the data retention policies of the popular LBS applications such as *Facebook Places*, *FourSquare*, *Loopt* and *Gowalla*, the LBS providers log all the LBS requests sent by the LBS users. This even includes some of the sensitive requests of the users (e.g. for the local campaign headquarter of a given political party or spiritual center for a given religion). And since there are no commercial or public anonymizers available to the users of LBS, the responsibility to preserve anonymity of the users lies with the LBS provider.

Thus in this scenario, the LBS provider needs to anonymize the request logs to provide sender k-anonymity against an attacker who can access the anonymized request

logs, all the past locations (trajectory) of the users and design of the system that provides the sender k-anonymity. We refer to this privacy guarantee as the Trajectory and Policy aware Sender k-anonymity.
The key challenges addressed in this study are :

[1] propose sender k-anonymity guarantee against an attacker who has access to the LBS requests log, user trajectory over a period of time and who knows the design of the system used to provide privacy.

[2] propose a method based on spatial cloaking to anonymize the LBS request logs to preserving the sender k-anonymity against the attacker who gains access to LBS requests, user trajectories and complete details of the anonymization.

[3] find the optimum privacy preserving anonymization for the LBS request logs.

The remainder of this thesis is organized as follows. In Chapter 3 we address the challenges of policy-aware sender k-anonymity for the requests issued at single instant of time. In Chapter 4 we address the above mentioned challenges of trajectory and policy aware sender k-anonymity for the log of LBS request. In Chapter 5 we discuss the legal requirements of privacy (or lack of it) in various countries for location-based services. In Chapter 6 we discuss some of the open problems in sender anonymity for Location-based services.

Chapter 3

Snapshot Policy-aware Sender k-anonymity

3.1 Introduction

Recent years have witnessed increased demand for Location-Based Services (LBS), which answer requests of mobile device users for services in their proximity (e.g. “find the nearest gas station”, “Thai restaurant”, “hospital”). While some such LBS providers are available in wireless networks since 2001 [exa], their proliferation has been limited, among other reasons, by privacy concerns.

In this chapter we address one such concern, which pertains to hiding the identity of the sender of more sensitive requests (e.g. for the local campaign headquarters of a given political party, spiritual center for a given religion, etc.) in order to keep her interests private. The sender’s identity must be protected even against attackers who, via hacking or subpoenas, gain access to *a)* the request (from the LBS provider’s log) and *b)* to the locations of the mobile user and other nearby users at the time of the request (from the wireless service provider) and *c)* who know the “design” of the system used to provide this protection. The assumption *c)* about the attackers is based on a well accepted principle of designing a private and secure system - “The design is not secret” [Sal74]. This is indeed a realistic threat since an attacker with subpoena powers (e.g. a federal agency) or a disgruntled ex-employee can obtain the “design” of the system.

In the context of LBS, the best-studied identity protection measure is known as *sender k-anonymity* [KGMP07], which is intended to guarantee that the request log and precise location information are insufficient to distinguish among the actual requester and k-1 other possible requesters. Typical sender anonymization algorithms [GG03, MCA06, KGMP07, XC07] are based on hiding the sender's precise location in the request, substituting instead a *cloak*, i.e. a region containing this location. The cloak is usually chosen from among regions of a pre-defined shape (circular, rectangular, etc.), to include at least k-1 other mobile users. To maximize the utility of the answer to the service request, usually the tightest cloak containing k users is picked. We refer to these utility-maximization policies of choosing cloaks as *k-inside*.

In the version of sender k-anonymity provided by the k-inside policies [GG03, MCA06, XC07], the principle that the design of a system is not secret is ignored. Therefore, such privacy guarantee does not hold against an attacker who knows the "design" i.e. the policy used to pick the cloaks for locations. The next example illustrates how an attacker who knows the cloaking policy ("policy-aware" attacker) can identify the sender when the cloaks are selected using a k-inside policy.

Example 3. *Figure 3.1 shows the 2-inside policy obtained using the algorithm described in [MCA06] for the location database of Table 3.1. The algorithm assumes a static quad-tree based partitioning of a geographic space and uses quadrants and sub-quadrants (combination of two adjacent quadrants) as possible cloaks. For a given requester, the algorithms picks the smallest cloak (containing the requester) that contains k-1 other users. For k=2 the algorithm cloaks* A *and* B *to R_1,* C *to R_3 and* S *and* T *to R_2. Since each of these cloaks contains at least 2 locations this is a 2-inside policy. Assume there is an attacker that has access to the location database D_1 (via hacking or subpoena) and is "design-aware" i.e. knows the 2-inside policy used to provide sender 2-anonymity. If this attacker observes an LBS service request with cloak R_3, he can identify the sender as C!.* □

The privacy guarantee of the k-inside policy have been refined by additional constraints such as *k-reciprocity*[KGMP07] and *k-sharing* [CM07]. We show (in Section 4.7) that since the policy-aware attacker is not considered, these additional constraints also fail to provide sender k-anonymity.

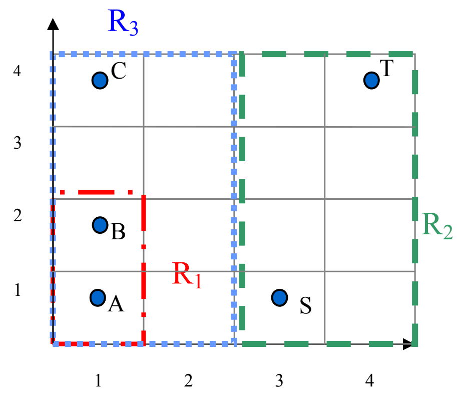


Figure 3.1: 2-inside policy

userid	locx	locy
. . .	. . .	. . .
Alice	1	1
Bob	1	2
Carol	1	4
Sam	3	1
Tom	4	4
. . .	. . .	. . .

Table 3.1: Location Database D_1

As described later in detail, to preserve sender k-anonymity against a policy-aware attacker, in some cases the cloak used for a location needs to be bigger (and include more than k locations) than the cloak picked by a k-inside policy. As a result, a cloaking that provides policy-aware sender k-anonymity (*policy-aware cloaking*) may have reduced utility in comparison to a k-inside policy. Moreover unlike k-inside policies where one can find the utility-maximizing (optimum) cloaking for each user by considering a small subset of all the users, for optimum policy-aware cloaking one has to consider all the users (as described later in Section 3.4), which is computationally more expensive than optimum k-inside policy. Hence policy-aware sender k-anonymity trades utility and performance for stronger privacy. In this chapter, we describe our findings on identifying the "sweet spot" in this tradeoff.

Our contributions. In addition to showing that k-inside policies achieve sender anonymity only against attackers who are policy-unaware, and is not proof against policy-aware attackers, our contributions include the following.

[1] We formalize the classes of policy-unaware and policy-aware attackers, and define a novel, stronger privacy guarantee: *sender anonymity against policy-aware attackers*. We prove formally that this guarantee strictly subsumes sender anonymity against policy-unaware attackers.

[2] We study the problem of finding, among all policy-aware sender k-anonymizations of a set of mobile users, one with optimum utility. We show that the problem of finding optimum policy-aware sender k-anonymity depends upon the type of cloaks used. In particular, we show that when the cloaks are circles whose centers are selected from a given set of points, the problem is NP-complete, but becomes PTIME for cloaks picked from among the quadrants of a quad-tree-based partition of the map (a common choice in state-of-the-art anonymization solutions [GG03, MCA06]).

[3] We implement and evaluate experimentally our optimum policy-aware anonymization algorithm. Even though finding optimum policy-aware anonymization is computationally costly in comparison to finding optimum k-inside policy (that uses the same cloak types) we show that our algorithm is practical and scales extremely well with the number of service requests: it takes less than 1 second to anonymize 250k requests from users in the San Francisco Bay area (using a single anonymization server) and can scale up to 1 million requests using 16 servers in parallel.

[4] As stated earlier, the policy-aware cloaking may result in some loss of utility in comparison to a k-inside policy (that uses the same cloak type). We show empirically that the utility reduction traded for the stronger privacy guarantee is reasonable: the average cloak area is at most 1.7 times the average area of the tightest cloaks used for policy-unaware anonymity.

Scope of our work. More recently, several extensions to sender k-anonymity has been proposed, such as allowing *user specified k* (in [GL05, CM07]) and defending against *trajectory-aware attacker* [BWJ05, XC07, CM07] where the attacker has knowledge of when multiple requests have originated from the same (a priori unknown) user, even if they are sent at different times and from different locations. While these extensions are important, our work improves upon the foundations of these extensions, namely make it policy-aware. We leave as future work the extension of the policy-aware sender k-anonymity to handle trajectory-awareness and user-specified k.

Paper outline. The remainder of the chapter is organized as follows. In Section 3.2, we show how we model an LBS. We define sender anonymity and the classes of attacks it defends against in Section 3.3. Section 3.4 gives our PTIME algorithm for finding a policy-aware anonymization of maximum utility. In Section 3.5 we describe how we utilize the inherent parallelism in the problem to obtain greater scalability and report on the experimental evaluation in Section 3.6. We discuss related work in Section 4.7 and conclude in Section 3.8.

3.2 LBS Models

This section introduces a basic model of providing location based services, based on information about user locations provided by a wireless network. It then describes modifications and additional components required for privacy support.

3.2.1 Basic LBS Model

Wireless "Communications Service Providers" (CSPs) can derive the approximate location of user devices, through a variety of mechanisms, including triangulation based on signal strength or time-delay to multiple cell towers, and GPS capabilities on the device. In the US, the E911 Requirement [E91] mandates that CSPs provide the necessary infrastructure to determine the location of mobile devices within a range of 50 to 300 meters, depending on the technology used. This infrastructure must be available when users call the emergency 911 number, but can also be used to support other services, including location-based information services. It is now common for CSPs to include specialized network components, which are called *Mobile Positioning Center* (MPC) in the CDMA standard, that serve as a logically centralized point that provides access to device locations for E911 and other location based services.

An abstract model is used here to study location-based services and their privacy characteristics. For simplicity, we model a geographic area as a 2-dimensional space and user's location as integer coordinates within this 2-dimensional space. There are four core elements in the delivery of a location-based service: the user making a request, typically called the *sender*, the (wireless) Communication Service Provider, denoted as *CSP*, the Mobile Positioning Center operated by the CSP, denoted as *MPC*, and the Location Based Service (LBS) provider, denoted as *LBS*. We view the CSP to be a trusted agent that operates the MPC. Although in practice the MPC provides the approximate location of each user's device, for simplicity we take the value produced by the MPC as the device's exact location. A sender's request for a location-based service is processed by the CSP, which obtains the user's location from the MPC and forwards the request to the LBS.

We abstract from the fact that location is usually determined only on demand, and assume in our investigation that the locations of all devices are eagerly computed and available. This eagerness assumption is appropriate in connection with the study of privacy guarantees, since we target attackers who might be able to reconstitute all device locations, perhaps by hacking in real-time, by hacking logs, or by subpoena-induced cooperation of the CSP.

Location Database. In the abstract model, for simplicity we assume that the device locations made available by the MPC are stored in a relational database, called the *location database*. (This database might be virtual.) Although its actual schema can vary from CSP to CSP, it is essentially equivalent to a single relation schema

$$\mathbf{D} = \{userid, locx, locy\}.$$

Here, the domains of attributes $locx$ and $locy$ are the domains of x and y coordinates in the 2-dimensional space used to model the geographic region.

In this investigation, we assume that the location database is updated periodically (e.g., every 30 seconds) to reflect the movement of users. Multiple location-based requests can be made against each snapshot. Thus the state of a location database over a period of time can be modeled as a sequence of different instances of schema **D**.

We represent the set of all possible instances of **D** by $\mathcal{D}$. An example instance $D_1 \in \mathcal{D}$ is shown in the Table 3.1, and illustrated diagrammatically in Figure 3.1.

The following definition allows us to focus on the precise information associated with a sender's request for a location-based service. (In the following section we describe how this request might be modified by the CSP to provide privacy protections before forwarding to the LBS.)

Definition 1. [**Service Request**] A *service request* is a tuple $\langle u, (x, y), V\rangle$ where u is a sender identifier, (x, y) are coordinates in 2-dimensional space and V is a vector of name-value pairs. We say that the service request is *valid* w.r.t a location database D if $\langle u, x, y\rangle \in D$.

Intuitively, the name-value pairs contain the categories and specifics of the sought services.

We define the function $id(SR)$ that returns the user id in the service request and another function $loc(SR)$ that returns the location co-ordinates (x, y).

Although a service request SR itself is created by the CSP, based on a request from a sender u, we sometimes refer to SR as having been sent by u.

Example 4. *The following are examples of service requests sent respectively by users* Alice*,* Bob*,* Carol*,* Sam*, and* Tom*:*

$SR_a = \langle \text{Alice}, (1, 1), [(poi, rest), (cat, ital)]\rangle,$

$SR_b = \langle \text{Bob}, (1, 2), [(poi, groc), (cat, asian)]\rangle,$

$SR_c = \langle \text{Carol}, (1, 4), [(poi, rest), (cat, ital)]\rangle,$

$SR_s = \langle \text{Sam}, (3, 1), [(poi, rest), (cat, ital)]\rangle,$

$SR_t = \langle \text{Tom}, (4, 4), [(poi, cinema), (cat, drama)]\rangle.$

(Here "poi" *stands for "point of interest",* "cat" *stands for "category",* "rest" *stands for "restaurant", etc.) All five service requests are valid w.r.t. the location database instance* D_1 *of Table 3.1.* □

In the abstract model, the service requests are created by the CSP using a combination of a request for information from a user, along with the user's location as provided by the MPC. We will therefore assume for our ongoing discussion that each service request is valid w.r.t. the current location database instance.

3.2.2 Privacy-conscious LBS Model

In realistic privacy solutions, the goal is not to hide information from everybody, but rather to minimize the number of parties one needs to trust to achieve the desired communication. The fundamental assumption underlying the privacy-conscious LBS model studied here is that the CSP is a trusted party, and nobody else is. In particular, the LBS is not trusted, reflecting the fact that it is usually a third-party provider that is not under the CSP's control.

We extend the basic LBS model of Subsection 3.2.1 to support mobile users in accessing the LBS without revealing their identity to anyone except for the CSP. Users rely on the CSP to ensure this goal. While we assume that the CSP can be trusted to perform the privacy-ensuring computations and not log them, we will assume that attackers may be able to obtain information about the locations of individual users at different times. (This might arise due to hacking, or to subpoenas, if the CSP is logging user locations for the purposes of advertising or service personalizations.) For the worst case, then, we assume that the sequence of location databases is available to the attacker.

In the privacy-conscious LBS model, a user sends a location-based request to the CSP over a channel that is assumed to be secure (this is typically the case in cell phone networks). The CSP constructs the corresponding service request SR, and based on this, will send a request on behalf of the user to an LBS L. The CSP cannot sent SR itself, because this includes the sender's identity, which would be revealed to both the LBS provider (which may be an attacker) and to any potential attackers listening on the channel. Also, simply removing the sender identity from SR does not suffice, because the identity can be obtained by examining the location database (which is assumed to be available to attackers). While there are many approaches to anonymize a request SR so that it does not reveal the requester's identity, for the current investigation we shall use the following classical one.

Definition 2. [**Anonymized Request**] An anonymized request is a tuple $\langle rid, \rho, V \rangle$ where rid is a unique request identifier, ρ is a connected, closed region in the plane, and V is a vector of name-value pairs. If ρ is a rectangular region with vertical and horizontal sides, then we also denote this anonymized request as

$$\langle rid, (x_1, y_1, x_2, y_2), V \rangle$$

where (x_1, y_1) ((x_2, y_2)) specifies the southwest (respectively northeast) corners of a rectangular region.

From now on we refer to these regions in anonymized requests as *cloaks*. We also define a function *reg(AR)* that returns the cloak from an anonymized request AR.

Example 5. *The following is a list of anonymized requests, whose cloaks are depicted in Figure 3.1.*

AR_a = $\langle 167, (0, 0, 1, 2), [(poi, rest), (cat, ital)] \rangle$
AR_b = $\langle 168, (0, 0, 1, 2), [(poi, groc), (cat, asian)] \rangle$
AR_c = $\langle 169, (0, 0, 2, 4), [(poi, rest), (cat, ital)] \rangle$
AR_s = $\langle 170, (2, 0, 4, 4), [(poi, rest), (cat, ital)] \rangle$
AR_t = $\langle 171, (2, 0, 4, 4), [(poi, cinema), (cat, drama)] \rangle$ □

Definition 3. We say that an anonymized request $AR = \langle id, \rho, V \rangle$ *masks* a service request $SR = \langle u', (x', y'), V' \rangle$ if the location (x', y') is an element of the cloak ρ and V=V'.

Example 6. *For each x in* $\{$*a, b, c, s, t*$\}$*, the anonymized request* AR_x *of Example 5 masks the service request* SR_x *of Example 4.* □

Instead of sending a service request SR to the LBS provider, the CSP forwards to the LBS provider an anonymized request that masks SR. The next section discusses how such anonymized requests are constructed.

3.3 Policy-aware k-anonymity

Prior research considers anonymization algorithms that cloak the sender's location with a region covering the location of $k - 1$ additional mobile users [GG03, MCA06, XC07]. The intention is that an attacker who observes the anonymized request and has access to the location database can not reduce the number of possible senders below k, since there are k potential service requests, one for each of the senders covered by the cloak of the anonymized request, that could have lead to the same anonymized request. While the cloaking algorithms proposed in the literature use different cloak shapes (quadrants [GG03, MCA06], minimum bounding circles [XC07], etc.) they agree in one important aspect: to maximize utility of the service, the tightest cloak that includes k users is picked. We term this class of cloaking policies as *k-inside* policies.

The results in this section are motivated by the observation (described in Section 3.1) that if the attacker is aware of the k-inside policy used by the CSP, then for some location databases he is able to reduce the number of possible senders below k, defeating the purpose of anonymization. We set out to defend against such "policy-aware" attackers (for general classes of cloaking policies, including but not limited to the k-inside policy). To this end we need to formalize the classes of policy-aware and -unaware attackers. In turn, this requires the formalization of the notion of cloaking policy, as the CSP's method of obtaining an anonymized request AR from a service request SR and a given location database instance D.

Definition 4. [**Policy**] A *policy* is a deterministic procedure P that takes as input a location database instance D and a service request SR, and outputs an anonymized request AR:

P : $\{$instances of location database$\} \times \{$service requests$\} \rightarrow \{$anonymized requests$\}$.

A policy P is *masking* if for every service request, the location specified in the service request lies within the cloak in the anonymized request it is mapped to. Formally,

$$\forall D\, \forall SR\ loc(SR) \in reg(P(D, SR)).$$

In this investigation we consider only masking policies, so from now on we use the term policy to refer to a masking policy.

Example 7. *Assume that the current location database instance is D_1, as shown in Table 3.1. The policy P_1 for the service requests shown in Example 4 is as follows:*

$P_1(D_1, SR_a) = AR_a \qquad P_1(D_1, SR_s) = AR_s$

$P_1(D_1, SR_b) = AR_b \qquad P_1(D_1, SR_t) = AR_t$

$P_1(D_1, SR_c) = AR_c$

where $AR_a, AR_b, AR_c, AR_s, AR_t$ are the anonymized requests of Example 5 and the cloaks used in these anonymized requests are shown in Figure 3.1 (where $reg(AR_a) = reg(AR_b) = R_1$, $reg(AR_c) = R_3$ and $reg(AR_s) = reg(AR_t) = R_2$). $\square$

The Attacker Model. We now proceed to formalizing the privacy guarantee of policy-aware sender k-anonymity. To this end, we need a framework for describing what an attacker knows about the policy being used by the CSP to anonymize requests. At the one extreme, an attacker may know exactly which policy is being used; as formally defined below these will be called "policy-aware" attackers. At the other extreme of interest here, an attacker may know only that the policy is based on the use of some family $\mathbf{C}$ of possible cloaking regions (e.g. rectangles, quadrants of a given quad tree, circles with center from a given set). Given such a family $\mathbf{C}$, we let $\mathbf{P_C}$ denote the set of all policies that use cloaking regions from $\mathbf{C}$. As formally defined below, attackers who know only that the policy used is an element of $\mathbf{P_C}$ for some set $\mathbf{C}$ will be called "policy-unaware (relative to $\mathbf{C}$)".

We target a strong, information-theoretic definition of privacy. To this end, we model attackers as a function taking certain input to launch the attack. There are no limiting assumptions on the computational resources expended to compute this function. The only assumptions are on what input the function has (intuitively, the information that the attacker sees). We classify this input into two groups as follows.

Design time: Even before the attacker observes any anonymized request, he may know

- the targeted level k of sender k-anonymity, and
- the family of candidate policies $\mathbf{P}$ (which in our study is typically either a singleton, or a set $\mathbf{P_C}$ for some family $\mathbf{C}$ of cloaking regions)

Run time: The attacker can observe (or reconstruct after the fact, via log hacking or subpoenas)

- the instance D of the location database (corresponding to a snapshot of all of the sender locations), and
- the set of anonymized requests made against this snapshot.

Notice that hacking attacks may be unable to reconstruct the entire location database. If sender k-anonymity is provided under the above assumptions, then it is also provided if the attacker has only partial knowledge of D.

The attack function models the following attack: starting from the observation of a set A of anonymized requests and the full knowledge of the location database D, the attacker "reverse engineers" the anonymized requests to obtain the possible service requests masked by A and compatible with the candidate policies in $\mathcal{P}$. We capture this result of the attack by defining the notion of *Possible Reverse Engineering (PRE)* of a set of anonymized requests.

Definition 5. [**Possible Reverse Engineering**] Consider a family of policies $\mathbf{P}$, a location database D and a set of anonymized requests $A = \{AR_i\}_{1 \leq i \leq n}$. A *Possible Reverse Engineering (PRE)* π of A w.r.t. D and $\mathbf{P}$ is a function from anonymized requests to service requests such that

- $\pi(AR_i)$ is valid w.r.t. D for all $1 \leq i \leq n$, and
- there exists some $P' \in \mathbf{P}$, such that $P'(D, \pi(AR_i)) = AR_i$ for each $1 \leq i \leq n$.

Intuitively, a PRE π associates with every anonymized request AR in A a possible service request that could have generated AR, based on some fixed policy P' from the family of candidates $\mathbf{P}$. We represent the set of all PREs of a set A w.r.t. D and $\mathbf{P}$ as $PRE(A, D, \mathbf{P})$.

Sender k-anonymity. We are now ready to define sender k-anonymity. Intuitively, this will capture the property that, even if the attacker uses the available information to flawlessly compute (no matter at what computational cost) all PREs of the observed set of anonymized requests, these PREs still point to at least k possible senders for each anonymized request. We consider it a breach of sender k-anonymity if the attacker succeeds in reducing the set of possible senders to fewer than k. We first define sender k-anonymity as a property of a set A of anonymized requests w.r.t. a location database D and a family of policies $\mathbf{P}$. Since the anonymized requests are obtained using a policy P, it is easy to extend the definition as a property of a policy P.

Definition 6. [**Sender k-Anonymity**] Let $\mathbf{P}$ be a family of policies and D a location database. Let A be a finite set of anonymized requests obtained using a policy $P \in \mathbf{P}$. We say that A provides *sender k-anonymity against* $\mathbf{P}$*-aware attackers on* D if there are PREs $\pi_1 \ldots \pi_k \in PRE(A, D, \mathbf{P})$ such that for each $AR \in A$ and each pair i, j satisfying $1 \leq i < j \leq k$, $id(\pi_i(AR)) \neq (id(\pi_j(AR))$.

We say that policy P provides *sender k-anonymity against* $\mathbf{P}$*-aware attackers on* D if for each finite set S of service requests (valid w.r.t. D), the set of anonymized requests $\{P(D, SR) \mid SR \in S\}$ provides sender k-anonymity against $\mathbf{P}$-aware attackers on D.

We say that P provides *sender k-anonymity against* $\mathbf{P}$*-aware attackers* if for every location database D, P provides sender k-anonymity against $\mathbf{P}$-aware attackers on D.

Since our attacker model is parameterized by the family of candidate policies $\mathcal{P}$ and the set of observed anonymized requests A, by varying these sets one can enumerate different classes of attackers and the corresponding flavors of sender anonymity. In this investigation we focus on two extremes.

- A *policy-unaware* attacker (relative to family $\mathbf{C}$ of possible cloaking regions) does not know which particular cloaking policy in $\mathbf{P_C}$ is used by the CSP, and observes only one anonymized request.
- A *policy-aware* attacker knows the specific policy P used by the CSP, and is able to observe and memorize all anonymized requests.

We show next that the class of policy-aware attackers is strictly more powerful (in terms of breaching sender k-anonymity) than the class of policy-unaware attackers.

Example 8. *Let's revisit Example 3 of Section 3.1. It can be easily observed that for the users of Table 3.1 the policy P_1 in Example 7 is based on the cloaking described in Example 3 When the policy-unaware attacker observes AR_c and tries to reverse engineer the service requests that could have generated it. He finds 3 PREs $\pi_1(AR_c) = SR_c$ and $\pi_2(AR_c) = \langle Alice, (1,1), [(poi, rest), (cat, ital)]\rangle$ and $\pi_3(AR_c) = \langle Bob, (1,2),$*

$[(poi, rest), (cat, ital)]\rangle$ with distinct users, Alice, Bob and Carrol. So policy P_1 provides sender 2-anonymity against policy-unaware attackers on D_1. In contrast, the $\{P_1\}$-aware attacker who observes AR_c can construct only one PRE, involving Carol, whose identity is completely compromised. Thus the $\{P_1\}$-aware attacker breaches sender 2-anonymity in a case when policy-unaware attacker cannot. □

In the remainder of the chapter we target an anonymization algorithm that preserves sender k-anonymity against the class of policy-aware attackers. Such an algorithm will also defend against policy-unaware attackers. This claim is formalized below (for proof see[DHVZ09]).

Proposition 1. *Let A be a set of anonymized requests obtained using policy P on location database D. If A provides sender k-anonymity against a policy-aware attacker on D, it also provides sender k-anonymity against a policy-unaware attacker on D.*

Sender k-Anonymity and k-inside Policies. We first check the privacy provided by some of the cloaking algorithms proposed in the literature [GG03, MCA06, XC07] against the two classes of attackers introduced above. As it turns out, they only defend against policy-unaware attackers.

Recall that all of these algorithms implement a k-inside cloaking policy, which we use to obtain generic results that hold for all algorithms in this class.

The following example shows a 2-inside policy that provides sender 2-anonymity against policy-unaware attackers.

Example 9. *For location database instance D_1 of Figure 3.1, the policy P_1 of Example 7 uses cloaks R_1, R_2 and R_3 to anonymize the service requests. Since each of these cloaks contains at least 2 locations, P_1 is a 2-inside policy. Now recall from Example 8 that the policy-unaware attacker could not reduce the set of possible senders of each request to less than 2.* □

We can show that the finding in Example 9 is not accidental:

Proposition 2. *A k-inside policy provides sender k-anonymity against policy-unaware attackers.*

In contrast, recall that Example 8 illustrates a case in which a *k-inside* policy does not provide sender k-anonymity against a policy-aware attacker, leading to the following claim.

Proposition 3. *Not all k-inside policies provide sender k-anonymity against policy-aware attackers.*

Since by Proposition 3, the prior anonymization algorithms do not satisfy our goal of defending against policy-aware attackers, we need to search for a novel algorithm.

Before presenting this algorithm in Section 3.4, we illustrate a policy that does provide sender 2-anonymity against policy-aware attackers.

Example 10. *For the location database instance D_1, we describe a policy P_2 for the service requests shown in Example 4:*

$P_2(D_1, SR_a) = \langle 167, R_3, [(poi, rest), (cat, ital)]\rangle$,

$P_2(D_1, SR_b) = \langle 168, R_3, [(poi, groc), (cat, asian)]\rangle$,

$P_2(D_1, SR_c) = \langle 169, R_3, [(poi, rest), (cat, ital)]\rangle$,

$P_2(D_1, SR_s) = \langle 170, R_2, [(poi, rest), (cat, ital)]\rangle$, *and*

$P_2(D_1, SR_t) = \langle 171, R_2, [(poi, rest), (cat, thai)]\rangle$.

The cloaks R_2 and R_3 used in these anonymized requests are those depicted in Figure 3.1. The readers can check that P_2 provides privacy against $\{P_2\}$-aware attackers since for each anonymized request one can construct 2 PREs using policy P_2. □

3.4 Optimal k-anonymity

For the same location database, there may exist several policies that provide policy-aware sender k-anonymity, raising the obvious question of which one to use. In this section we address the problem of finding the policy of highest utility to the users. Prior work on policy-unaware anonymity proposes that one way to maximize utility is to minimize the cloak area. A smaller cloak allows for more efficient processing of range queries (e.g. find gas stations within 2 miles) at the LBS as well as more efficient filtering of results at clients. Since we don't know a priori the users who are going to send a request at a given snapshot of location database, we compare policies for the case when every user sends a request.

Cost of a policy. We introduce the cost of a policy to quantitatively capture the fact that the utility is maximized as the cloak area is minimized. We define the *cost of an anonymized request* AR as the area of its cloak $reg(AR)$. Given a location database D and a set S of service requests valid w.r.t. D, the *cost of* S *under* P is defined as $\sum_{SR \in S}$ cost of $P(D, SR)$. The *cost of a policy* P *on* D, denoted $Cost(P, D)$, is computed as the cost of the set of service requests obtained if every user in D issues precisely one request (of immaterial parameters); it is the cost of the set of service requests

$$S = \{\langle u, (x, y), V\rangle | \, (u, x, y) \in D\}$$

where V is some arbitrary vector of name-value pairs.

Optimal policy. We next focus on the problem of obtaining, for a given location database D, an optimal (cost-minimal) policy that provides sender k-anonymity against policy-aware attackers. We show that the complexity of this problem depends upon the type of cloaks used in the anonymization. In particular we show that the problem is NP-complete if the cloaks are of circular shape, and are picked by choosing the center from a given set of points (e.g. public landmarks such as libraries, train stations or cell towers) and by choosing the radius freely. It comes therefore as a pleasant surprise that the problem becomes PTIME for a version in which cloaks are picked among the quadrants of a quad-tree-based partition of the map.

We first detail the circular-cloak version of the problem. Let D be an instance of location database and SC be a set of possible centers. Find a policy-aware sender k-anonymous policy P that minimizes $Cost(P, D)$. P uses circular cloaks, each centered at some point from SC, with no restriction on the radius. We call this problem *Optimal Policy-aware Bulk-anonymization with Circular cloaks*. We find the following negative result.

Theorem 1. *Optimal Policy-aware Bulk-anonymization with Circular cloaks is NP-Complete.*

Note that the NP-completeness is in the size of the location database, meaning that optimal policy-aware anonymization is practically infeasible.

There is good news, however, if we consider a different type of cloaks from which the policy may choose. This result has high practical impact, since the cloak type in question is already widely used in the literature. We consider cloaks picked from among the quadrants corresponding to a quad-tree-based partitioning of a planar area. The quad tree is a well-known structure for organizing spatial data, and it has been used in a number of anonymization solutions [GG03, MCA06]. As the name suggests, it is a tree in which every non-leaf node has exactly 4 child nodes. In a quad tree representation of a (square shaped) map, the root node represents the entire map. The region is then divided equally into 4 non-overlapping square quadrants, each of whom represents a child node of the root. Each quadrant is then again divided into 4 equal sub-quadrants that correspond to grandchildren of the root. This four-way splitting goes on until the desired level of granularity for the minimum region is reached.

A policy that anonymizes locations to cloaks represented by nodes of the quad-tree representation of a given map is known as *quad-tree* policy.

This brings us to our main finding.

Theorem 2. *An optimal quad-tree policy providing sender k-anonymity against policy-aware attackers can be found in PTIME.*

In the remainder of this section, we describe a PTIME algorithm to find an optimal quad-tree policy.

3.4.1 Reducing the Policy Search Space

Given a map with its associated quad tree T, and a location database D, it is easy to see that the space of all quad-tree policies cloaking locations in D by nodes in T is exponential in the size of D. This rules out solutions based on enumerating all policies.

Intuitively, the following key observation reduces the search space to polynomial size: both the property of being policy-aware sender k-anonymous, and the cost of the policy depend only on *how many* locations are cloaked by each node N of the quad tree T, being indifferent to *which* particular locations are cloaked by N. Calling policies equivalent if every quad tree node cloaks the same number of locations under both policies, one need not enumerate individual policies, enumerating policy equivalence classes instead. It turns out that only polynomially many such classes need to be inspected.

Equivalent Policies. We formalize this intuition next. Given location database D and quad tree T, two policies are *equivalent under* D, T if every node N of T cloaks the same number of locations from D under both policies. When D and T are clear from the context, we may not mention them. The following justifies why we need not discriminate among equivalent policies.

Lemma 1. If policies P_1, P_2 are equivalent under D, T, then

(a) P_1 and P_2 have the same cost; and

(b) P_1 provides policy-aware sender k-anonymity on D if and only if so does P_2.

3.4.2 A First-Cut Algorithm

For simplicity of exposition, we start by presenting a first-cut PTIME algorithm derived in the most direct way from the insight that equivalence classes suffice. We leave its optimization to Section 3.5.

Configurations. The first-cut algorithm manipulates equivalence classes of policies. It represents an equivalence class by keeping track for each quad tree node of the number of locations it cloaks. For technical convenience, this is done by equivalently tracking for each node N the number of locations that are located within N yet are *not* cloaked by N or any of its descendants. It is easy to translate between the two equivalence class representations.

Definition 7. [**Configuration**] Let D be a location database, and T be a quad tree rooted at node r. Let $d(m)$ denote the total number of locations from D that occur in the quadrant m. A *configuration* C of T is a function from the nodes of T to natural numbers, such that

(i) for every leaf node m in T, $C(m) \leq d(m)$; and

(ii) for every internal node m, $C(m) \leq \sum_{i=1}^{4} C(m_i)$,
where $m_1 \ldots m_4$ are the children of m.

We say that C is *complete* if $C(r) = 0$.

Item (i) of Definition 16 simply states that a node can be used to cloak at most as many locations as contained in its quadrant (since we only consider masking policies). Item (ii) states that the number of locations not cloaked by m's children is higher than the number of locations not cloaked by m, which is an immediate consequence of the fact that each child quadrant is contained in its parent.

Note that, given a policy, location database D and quad tree T, and a configuration C for T, we can exhibit in linear time one of the policies C represents (by arbitrarily selecting the $C(m)$ locations for each node m). The first-cut algorithm's strategy is to find a minimum-cost configuration, then exhibit (in linear time) one of the policies represented by it, picked nondeterministically.[1] Note that a configuration is exponentially more succinct than an explicit listing of the policies it represents; if we focus on any node m alone, there are exponentially many ways to pick $C(m)$ locations among those occurring in m.

Before explaining how the desired configuration is found, we address two technical issues that need to be solved to allow the first-cut algorithm to manipulate configurations without materializing policies (except for outputting the result). First, how to compute the cost of the represented policies without materializing them. Second, how to check if a configuration corresponds to policy-aware sender k-anonymous policies.

Computing Cost from Configurations. We define the cost of a configuration as the cost of the policies it represents (uniquely defined by Lemma 1(a)). This cost can be computed without materializing any represented policy, using the following function.

Definition 8. [**Configuration Cost**] Let D be a location database instance and C be a

[1]For the sake of conciseness, in the following we overload the term policy to denote functions from user locations to cloaks (instead of from service requests to anonymized requests as in Definition 4). The two notions of policy are inter-reducible.

configuration of a quad-tree T. We define the *cost of C on D*, denoted $Cost_c(C, D)$, as

$$Cost_c(C, D) := \sum_{n \in nodes(T)} f(n, C)$$

where $f(n, C)$ is given by

$$f(n, C) = \begin{cases} (d(n) - C(n)) \times area(n), & \text{n is leaf} \\ ((\sum_{i=1}^{4} C(n_i)) - C(n)) \times area(n), & \text{n is internal} \end{cases}$$

where $n_1 \dots n_4$ are the children of n and area(n) is the area of the quadrant corresponding to quad node n.

We can show that the configuration cost is precisely the cost of the represented policies:

Lemma 2. *Given a location database D, a quad tree T, a quad-tree policy P based on T and a configuration C representing P's equivalence class, we have $Cost_c(C, D) = Cost(P, D)$.*

Checking Sender Anonymity from Configurations. We turn to checking if the policies in the equivalence class represented by a given configuration are policy-aware sender k-anonymous, without materializing them. By Lemma 1(b), either all represented policies qualify, or none does. It turns out that it suffices to check directly that the configuration satisfies a property we call *k-summation*.

Definition 9. [**k-summation**] Let D be a location database instance and C a configuration of a quad tree T rooted at n. C satisfies *k-summation* if

- for a leaf node m
  - (i) if $d(m) < k$, then $C(m) = d(m)$.
  - (ii) if $d(m) \geq k$, then either $C(m) = d(m)$ or $C(m) \leq (d(m) - k)$.
- for an internal node m let $\Delta = \sum_{i=1}^{4} C(m_i)$, where $m_1 \dots m_4$ are the children of m

(iii) if $\Delta < k$, then $C(m) = \Delta$.

(iv) if $\Delta \geq k$, then either $C(m) = \Delta$ or $C(m) \leq (\Delta - k)$.

Intuitively, in Definition 18, clause (i) states that if node m's quadrant contains less than k locations, none of them can be cloaked by m lest k-anonymity be compromised. The cloaking responsibility for all $d(m)$ of them is "passed up" to m's ancestors ($C(m) = d(m)$). By clause (ii), if there are at least k locations, then either all of them are passed up, or at most $d(m) - k$ (since at least k must be cloaked together to preserve k-anonymity). Δ represents the number of locations whose cloaking responsibility is passed up from m's children to m. If there are too few of them (less than k) then they cannot be cloaked by m, who in turn passes the responsibility to its ancestors (in clause (iii)). Otherwise, m has the choice of either cloaking none of them ($C(m) = \Delta$ in clause (iv)), or cloaking at least k and passing up at most $\Delta - k$.

Lemma 3. Let T be the quad-tree representation of a map and D be an instance of the location database for that map. If C is a configuration of T and P a policy in the equivalence class C represents, then P is policy-aware k-anonymous on D if and only if C satisfies the k-summation property.

Algorithm $Bulk_{dp}$. Lemmas 2 and 3 justify an algorithm that explores the space of configurations satisfying k-summation, in search for a complete minimum-cost configuration under $Cost_c$.

The exploration is carried out as follows. Recall from Definition 8 that the cost of configuration C of a quad tree T rooted at m depends only on the number $C(m)$ of locations not cloaked by T's nodes, and is independent of the cloaking at the nodes outside of T. For this reason, it suffices if the search space includes, for every quad tree node m, all possible numbers u of locations whose cloaking responsibility is passed up to m's ancestors. That is, all possible values u for $C(m)$. For each such pair (m, u), the minimum cost is computed among all possible configurations C' of T with $C'(m) = u$. To this end, the algorithm considers all possible counts $u_1, \ldots, u_4$ of locations passed up to m by its children $m_1, \ldots, m_4$, and recursively computes the corresponding minimum cost for each (m_i, u_i) pair.

Redundant cost re-computation for m, u pairs is avoided by storing the result in the corresponding cell of a bi-dimensional matrix M indexed by quad tree nodes and by values for u. To enable the easy retrieval of the min-cost configuration from M, the entries for node m carry, besides the minimum cost, some bookkeeping information relating to the configurations at the children of m.

This yields the following dynamic programming algorithm $Bulk_{dp}$ that, given quad tree T and location database D, fills in a *configuration matrix* M of dimension $|T| \times |D|$, where $|T|$ denotes the number of nodes in T and $|D|$ the number of locations in D. Each entry $M[m][u]$ in the matrix is a tuple of the form $\langle x, u_1, u_2, u_3, u_4 \rangle$, pertaining to a configuration C for the quad sub-tree rooted at m, such that $C(m) = u$, $Cost_c(C, D) = x$, and $C(m_i) = u_i$ where $m_1, \ldots, m_4$ are the children of m. The algorithm traverses the quad-tree T bottom-up starting from its leaf nodes, and for each node m and $1 \leq u \leq n$ fills in the entry $M[m][u]$ using the rows for $m_1, \ldots, m_4$.

Notice that it is easy to retrieve in polynomial time a minimum-cost complete configuration from M, by a top-down traversal of T. First, pick a minimum-cost entry in the row corresponding to the root of T. This entry lists for each child m_i of the root the value $C(m_i) = u_i$ leading to the minimum cost. Now inspect for each m_i the corresponding row in M, picking again a minimum-cost entry, and continue recursively until all leaf nodes are reached.

Function $F(m)$ in line 16 limits the possibilities of the number of locations whose cloaking can be passed up by m. Notice that it rules out the values $d(m) - k + 1$ through $d(m) - 1$ since these imply cloaking less than k locations at m, which would immediately compromise k-anonymity. Quantity x is the minimum cost among all configurations C with k-summation for which $C(m) = u$. This is computed from the costs of the configurations at the 4 children, and the term $area(m) \times ((\sum_{l=1}^{4} u_l) - u)$, where $(\sum_{l=1}^{4} u_l) - u$ is the number of locations actually cloaked by m. Recall that the cost is found in the first component of the tuple stored in the matrix entry, whence the need for the projection operation M^1.

Notice how the algorithm mirrors the definition of the k-summation property (Definition 18) to ensure that only configurations satisfying k-summation are considered. By Lemma 3, these configurations represent only policy-aware sender k-anonymous policies. For instance, line 8 corresponds to case (i) in Definition 18, which

Algorithm 1 $Bulk_{dp}$

```
 1: for $1 \leq m \leq |T|$ do
 2:     for $1 \leq u \leq |D|$ do
 3:         M[m][u] := $\langle \infty, 0, 0, 0, 0 \rangle$ {initialize}
 4:     end for
 5: end for
 6: for all node $m \in T$ do
 7:     if (m is a leaf node) and (d(m) $<$ k) then
 8:         M[m][d(m)] := $\langle 0, 0, 0, 0, 0 \rangle$
 9:     else if (m is a leaf node) and (d(m) $\geq$ k) then
10:         M[m][d(m)] := $\langle 0, 0, 0, 0, 0 \rangle$
11:         for $0 \leq u \leq d(m) - k$ do
12:             M[m][u] := $\langle area(m) \times (d(m) - u), 0, 0, 0, 0 \rangle$
13:         end for
14:     else {m is a non-leaf node}
15:         let $m_1, m_2, m_3, m_4$ are children of m
16:         for all u in F(m) do
17:             pick $u_1 \in F(m_1), u_2 \in F(m_2), u_3 \in F(m_3)$,
18:             $u_4 \in F(m_4)$ that minimize the quantity
19:             $x := \sum_{l=1}^{4} M^1[m_l][u_l] + (area(m) \times ((\sum_{l=1}^{4} u_l) - u))$
20:             where
                $F(m)$ denotes the set $[0..(d(m) - k)] \cup \{d(m)\}$,
                and $M^1[i][j]$ returns the first component of the
                tuple at $M[i][j]$
21:             M[m][u] := $\langle x, u_1, u_2, u_3, u_4 \rangle$
22:         end for
23:     end if
24: end for
25: return M
```

prescribes that no locations are to be cloaked by m (all $d(m)$ locations occurring in its quadrant are passed up, $C(m) = d(m)$). Thus by Definition 8, the resulting cost is 0, which is what line 8 fills into the first component of $M[m][d(m)]$. Similarly, line 10 gives the cost corresponding to the case in the first disjunct of line (ii) of Definition 18; line 12 corresponds to the second disjunct. It's easy to see that:

Lemma 4. *Algorithm $Bulk_{dp}$ computes in each $M[m][u] = \langle x, u_1, u_2, u_3, u_4 \rangle$ the minimum configuration cost x among all configurations C with k-summation where $C(m) = u$ and where $C(m_i) = u_i$, with $m_1, \ldots, m_4$ the children of m.*

By the above discussion, the information in M suffices to retrieve in PTIME a minimum-cost configuration.

Complexity analysis. The running time of Algorithm $Bulk_{dp}$ is dominated by steps 16-18 , which, for internal node m, ranges each of u, u_1, u_2, u_3, u_4 over at most $|D|$ values (since $F(n) \leq d(n) \leq |D|$ for every n), resulting in $O(|D|^5)$ iterations. Summing up over all nodes m of the quad-tree, we obtain the complexity of $Bulk_{dp}$ in $(O|T||D|^5)$. Lemma 4 and this complexity analysis directly imply Theorem 2. While polynomial, and thus a welcome surprise in contrast to Theorem 1, the degree 5 is impractically high given the large size of the location database, which is why we consider optimizations in Section 3.5

Incremental Maintenance of M. Algorithm $Bulk_{dp}$ computes the optimal policy for a snapshot D of the location database starting from scratch (hence the name). As the users of the mobile network move around, the location database snapshot changes from D to D' at the next snapshot. Any optimal policy computed at snapshot D may not remain optimal for D', or may not provide policy-aware sender k-anonymity to users in D'. One can simply re-compute the optimal policy from scratch, calling algorithm $Bulk_{dp}$ on D' and T. Alternatively, if there are a large number of users in D but only few of them move between consecutive snapshots it makes sense to consider *incremental re-computation* of the optimal configuration matrix for D' starting from the optimal configuration matrix for D. This is easily accomplished by running the same bottom-up steps as algorithm $Bulk_{dp}$, with the added twist that the algorithm starts only from the quad tree leaves m

whose quadrants now contain a changed number $d(m)$ of locations.

3.5 Optimizations

While the first-cut algorithm $Bulk_{dp}$ is polynomial, it is far from practical yet, since a degree of 5 is prohibitive given the typical sizes of location databases (the location database of a wireless service provider in the San Francisco Bay may contain about one million users). In this section, we describe a series of optimizations of the naive algorithm to achieve practical running time, while guaranteeing to preserving the optimal cost.

From Quad to Binary Trees. The algorithm described in [GG03] pioneers the idea of using quadrants of a quad-tree as cloaks. In a quad-tree, if cloaking a location to a node does not provide the desired k-anonymity, the next possible option is the parent node. Since the parent node is 4 times the size of a child node, the granularity of cost increase is large. The cloaking policy in Casper [MCA06] reduces this granularity by considering *semi-quadrants* as cloaks (where a semi-quadrant is obtained by splitting a quadrant into two rectangles, either vertically or horizontally). The cloaks obtained in this approach are never larger than the cloaks obtained with the original quad-tree, and on average the cloak size is reduced.

While [MCA06] uses this idea to improve utility, we additionally exploit it here to improve running time. We too allow cloaks to be chosen among both the original quadrants of quad tree T, and their semi-quadrants. To this end we define a modified tree whose nodes are of either shape. It is a *binary tree* B obtained from T as follows. For each node m in T, let its 4 children in T be $m_{NW}, m_{SW}, m_{SE}, m_{NE}$, where the subscript gives their location (Southeast, etc.) in m. We divide m into two vertical semi-quadrants (rectangular) s_W in the West and s_E in the East. In B, m becomes the parent of s_W and s_E, s_W the parent of m_{NW}, m_{SW}, and s_E the parent of m_{SE}, m_{NE}. Notice that each node in B has only 2 children, each non-leaf quadrant node is a parent of two semi-quadrants, and each non-leaf semi-quadrant is a parent of two quadrants. Casper chooses between vertical or horizontal sub-quadrants at run-time, while for simplicity we statically partition quadrants into vertical semi-quadrants only.

We adapt the $Bulk_{dp}$ algorithm to this binary tree. The only change required is in step 5 of the algorithm. When computing each entry M[m][u] of the optimum configuration matrix we have to iterate through the configurations of two children only (compared to four in $Bulk_{dp}$). This reduces the complexity of the loop to $O(|D|^2)$ from $O(|D|^4)$, and that of the algorithm to $O(|B||D|^3)$.

Note that the cost of the optimal binary-tree based policy for a given location database instance may be different from the optimal cost of the original quad-tree based policy. If the size of a leaf node is kept the same in the binary tree and quad tree then the binary tree will need to have twice the height of the quad-tree to cover the same region. If k is also kept the same, than the cost of an optimal binary tree based policy is not more than the cost of an optimal quad tree policy, since any policy-aware anonymous quad tree policy is also a policy-aware anonymous policy for the binary tree. The remaining optimizations in this paper focus on the binary tree.

Pruning Suboptimal Configurations. For any node *m* of the binary tree, in the for loop of step 5, $Bulk_{dp}$ inspects $(d(m) - k + 1)$ sub-tree configurations (all possible configurations that satisfy k-summation) for the sub-tree rooted at *m*. We realize that some of these configurations need not be considered, as they are guaranteed to be sub-optimal. In fact we claim the following lemma:

Lemma 5. For a node m with height $h(m)$ (where the height of the root is 0), any configuration in which m passes up to its ancestors the cloaking responsibility for more than $(k + 1)h(m)$ but less than $d(m)$ locations, is not optimal.

By Lemma 5, it suffices to compute $(k + 1)h(m)$ configurations, by simply replacing function F in step 5 of algorithm $Bulk_{dp}$ with function $F'(m) = [0..((k + 1)h(m))] \cup \{d(m)\}$.

With this insight, the number of columns required in the optimum configuration matrix M becomes at most kh, where h is the height of the tree. In step 5, for a non-leaf node m, the algorithm computes $O(kh)$ configurations and to compute each such configuration, the "pick" action iterates over $O(kh)$ configurations of m's two children. This leads to a new upper bound of the overall running time, $O(|B|(kh)^3)$.

Note that if we fix a minimum area corresponding to the leaves of the tree, the height depends only on the area of the covered map, and, remarkably, we find a complexity upper bound that is independent of the size of the location database! However, this upper bound is only of theoretical interest, since we actually implement yet another optimization: we do not eagerly materialize all nodes of the binary tree. Instead, we split a (semi-)quadrant only if it contains sufficient users to maintain anonymity. In our experiments, we observed that the number of materialized nodes $|B|$ *does* depend on the size of the location database.

Precomputation. Again we focus on the FOR loop in step 5 of the $Bulk_{dp}$ algorithm. We observe that, across iterations of the loop, the "pick" command will repeatedly inspect certain configurations of m's children. For example, if one iteration works on the M entry for (m, u), inspecting for instance (m_1, u_1) and (m_2, u_2) such that $u_1 + u_2 = u$ for some v, then the next iteration $(m, u+1)$ will inspect the cases (m_1, u_1+1), (m_2, u_2) and (m_1, u_1), $(m_2, u_2 + 1)$, among others. The idea is to reuse this computation across iterations. To this end, we stage the computation in 2 parts. In the first stage we iterate over the $O(kh)$ configurations of both children to compute a temporary matrix $temp$. An entry $temp[m][j]$ in this matrix stores the minimum cost c of having $j = l_1 + l_2$ un-anonymized locations in the two children m_1 and m_2 of m (with l_1 in m_1 and l_2 in m_2).

$$temp[m][j] := \min_{l_1+l_2=j} \{M[m_1][l_1] + M[m_2][l_2]\} .$$

There are $O(kh)$ entries in this matrix and the complexity of this stage is bounded by $O((kh)^2)$. In the second stage, we create $O(kh)$ configurations using the $O(kh)$ entries of temp.

$$x := \min_{j=i \text{ or } j\geq i+k} \{temp[m][j] + (j - i) \times area(m)\} .$$

Thus the running time for the second stage is also bounded by $O((kh)^2)$. Therefore the overall complexity of the modified step 5 is $O((kh)^2)$ and the overall complexity of the algorithm is $O(|B|(kh)^2)$.

Runtime Prunning. We add further pruning to the optimized $O(m(kh)^2)$ version of our algorithm. In the second step of above mentioned optimization, for every non-leaf node m, we iterate through the temporary matrix b to find the cost c of having i un-anonymized locations in m, for 1 ≤ i ≤ hk. For a given value of i, as we iterate through matrix b to find the minimum cost, i.e. for 1 ≤ j ≤ hk, the value of the expression $(j - i)cost(m)$ increases. If $(j - i)cost(m)$ becomes greater than the current minimum we can break the iteration for that i since for every successive value of j, value of $(j - i)cost(m)$ will remain greater than current minimum (since and $b[m][j]$ is always non-negative). Because of the this pruning some $b[m][j]$'s are never used in computing any of the $M[m][i]$ therefore we compute the matrix b lazily.

Complexity Analysis in terms of $|D|$. Our complexity analysis so far was carried out for precision in terms of the size and height of the quad tree. While a gross upper bound for $|B|$ and h is $|D|$, leading to cubic running time in $|D|$, the real values of h and B depend on the skew of the locations in D. For instance, if the location distribution is uniform, it follows that $|B| \in O(\frac{|D|}{k})$ and $h \in O(\log(\frac{|D|}{k}))$, and the overall running time becomes $O(k|D|\log^2(\frac{|D|}{k}))$, i.e. linear for practical purposes in both k and $|D|$.

It turns out that this analysis is highly robust to relaxing the assumption on uniformity. Our experiments in Section 3.6.1 confirm the above formula even for realistic data whose distribution is quite skewed from uniform: 1.75 million locations reflecting the actual population density in the entire San Francisco Bay Area. The only examples we could create to force non-linear behavior are contrived.

Parallel Anonymization. We next explore a powerful technique for scaling the anonymization algorithm to cover large areas. The result is based on the key observation that the spatial nature of the problem features inherent parallelism that is easily exploited: just partition the region into sub-regions, putting each under the jurisdiction of an independent anonymization server. The servers run in parallel, each maintaining their own binary tree and location database, and seeing only requests issued in their jurisdiction. The policy in this distributed setting is a master policy which anonymizes a location l by referring to the policy constructed by the individual server under whose jurisdiction l falls.

One concern is that the obtained anonymization cost may no longer be optimal.

To see why, consider cases when the best way to anonymize location l by server 1 is to issue a cloak that spans the jurisdiction of server 1 and its neighboring server 2. Since server 1 does not have access to the requests and location database in server 2, it will use a different, larger cloak, completely contained within its own jurisdiction. However these cases occur only on the border of jurisdictions, and the case when the spanning cloak is unavoidable requires very low population density at the borders. We therefore expect only a minimal divergence from the optimal cost. We verify this expectation experimentally in Section 3.6.5, showing that the system throughput can be effectively increased as more servers are added, while the cost remains within 1% of the optimum.

Assuming a fixed pool of servers, we would like to partition the map into jurisdiction so as to balance server load (the number of locations per server). We show that this is satisfactorily achieved even by an unsophisticated partitioning scheme. We adopt a greedy scheme which, given a location database and a binary tree B, picks jurisdictions from among the nodes of B. The greedy partitioning algorithm starts with the root as the only jurisdiction in the list L. At every step, the algorithm picks a node from the list, all of whose children have either 0 or at least k locations. If multiple such nodes exist, pick the one with the higher number of locations. The node is then replaced in L with its children. This repeats until the size of the list reaches the desired number of servers. Intuitively, we first greedily split the nodes with the highest number of locations, to balance the number of locations falling in each jurisdiction.

Our experiments reported in the next Section 3.6 uncover the potential of parallel anonymization solutions. Note that in this work we do not advocate re-partitioning the map upon every location database snapshot, but instead picking a few representative snapshots and performing a static partition for each. The representatives pertain to various times of the day such as rush hours, night time, business hours, etc. In future work, we will study the systems issues related to the dynamic maintenance (and load re-balancing) of the server pool for highly dynamic fluctuations of the population density.

3.6 Experiments

We next verify experimentally that our optimized algorithm scales well with the size of the location database, and that the stronger privacy guarantee comes at a reasonable cost increase.

Platform. All our experiments were performed on an Intel Pentium4 3.20GHz machine running Linux with 2GB memory.

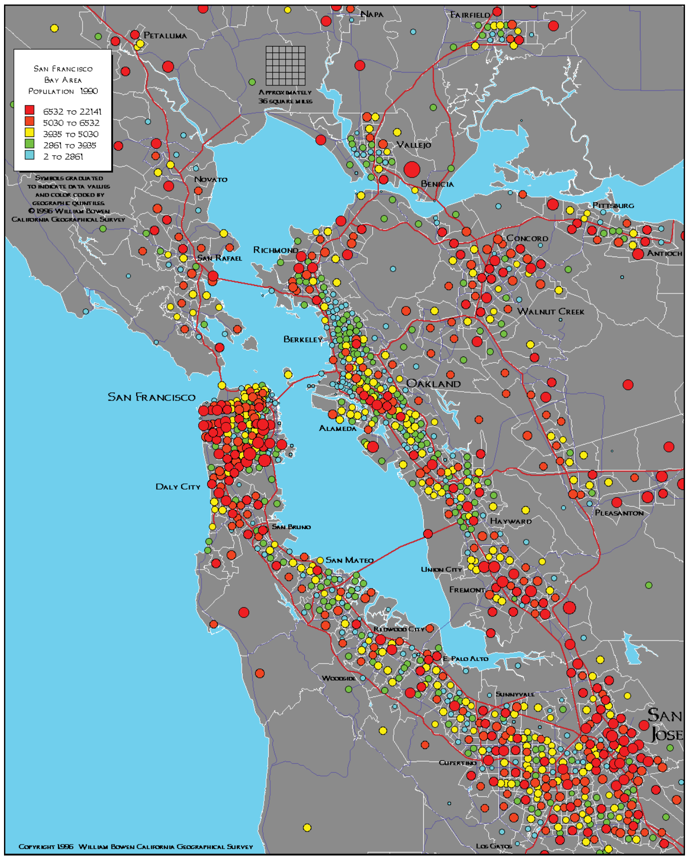


Figure 3.2: Population Density

Location Data. We set out to generate location data starting from a real-life map, using a real distribution of population density. Figure 3.2 illustrates the population density for the San Francisco Bay area in 1990, and is available from [Bow]. Unfortunately, the actual data values are not available, which is why we generated them as follows.

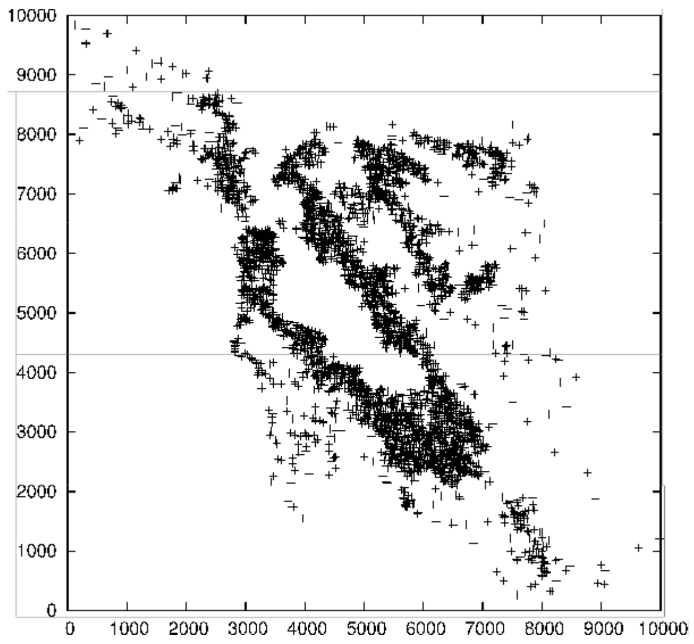
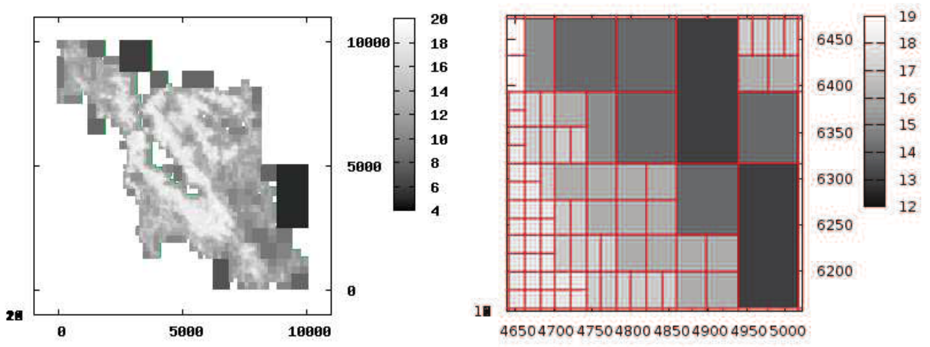


Figure 3.3: 5000 Street Intersections

We obtained a data set of street intersections in the same region (available at [Bria]). This dataset contains about 175k street intersection points. We conjectured that the population density is highly correlated with the intersection density. We validated this conjecture by plotting a random sample of 5000 points from this dataset (shown in Figure 3.3), and observing that it is roughly similar to the actual population density graph of Figure 3.2.

We inserted 10 locations around each intersection using a Gaussian distribution with standard deviation of 500 meters. We obtained a *Master* dataset of 1.75 million locations. We believe this number to be realistic, since although the total population of the San Francisco Bay area is around 7M, it corresponds to the maximum market share at national level for any single national wireless provider (according to the statistics published in [sta]). To scale the size of the location database for our experiments, we draw random samples of increasing sizes (100k, 200k etc.) from the Master data set.

Warm-up experiment: shape of the quad tree. Recall from Section 3.5 that the binary tree is not computed eagerly; we split a (semi-)quadrant only if the resulting children contain sufficient users to maintain anonymity. Figure 3.4(a) illustrates the tree



(a) Complete view (b) Zoomed-in view

Figure 3.4: Tree structure built on 1M data

structure built on the 1M sample of the master file with $k = 50$. It plots the quadrants and semi-quadrants in a 2-dimensional plane. The height information is presented by the gray scale, so that nodes of greater height are brighter. It turns out that a binary tree of maximum height 20 suffices to cover 1M locations, with no leaves containing more than 50 locations. Even when growing $|D|$ to 1.75M locations, the height of the tree never reaches 25. As expected, the denser areas lead to greater height, showing that the algorithm exploits the larger density to materialize finer-grained (semi-)quadrants, which in turn lead to smaller cloaks and better utility. Figure 3.4(b) gives a zoomed-in view at a portion of the map, illustrating the variation of leaf (semi-)quadrant sizes as a function of population density.

3.6.1 Bulk Anonymization Time

In this experiment we evaluate the running time of our algorithm (the optimized version) varying the size of the location databases, the anonymization degree k, and the number of anonymization servers. Figure 3.5(a) shows, for fixed $k = 50$, the running time for computing an optimum configuration with increasing location database size, with one curve per number of servers. The horizontal axis shows the number of locations while the vertical axis shows the time in seconds. The running time is linear in the number of locations $|D|$ for up to 1.75M locations, as predicted by the complexity analysis at the end of Section 3.4. Notice that 16 servers can bulk anonymize 1.75M locations in less than 1s, and 32 servers in less than 0.5s. We note that this is extremely good scalability, especially since our experiment stress-tests the algorithm to sizes of the location database that far exceed the ones reported in prior work on policy-unaware sender anonymity: at most 300K in [KGMP07].

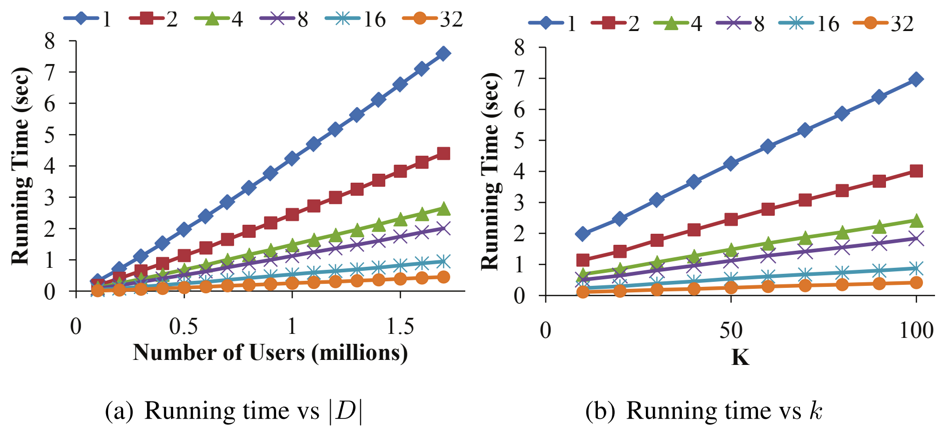


Figure 3.5: Linear running time in $|D|$ and k

Next, we inspect how the running time scales with k, keeping the number of locations fixed at 1M. Figure 3.5(b) shows that the time increases quasi-linearly (really sub-linearly) with k, again as predicted by the analysis at the end of Section 3.4.

3.6.2 Cost Overhead of Stronger Privacy

We expect that the stronger privacy guarantee will result in higher cost, by requiring larger cloaks in the anonymized requests. To evaluate the increase in cost from policy-unaware to policy-aware sender k-anonymity, we compared the $Cost$ (in Definition 3.4) of the optimum policy-aware sender k-anonymous policy obtained using our algorithm, with that of

- Casper: since it is the state-of-the-art policy-unaware anonymizing system based on semi-quadrants [MCA06], and our binary tree optimization was inspired by it.
- Optimum policy-unaware binary tree (PUB): since it uses the same type of cloak as our algorithm and a comparison would give a good measure of the penalty of stronger privacy.

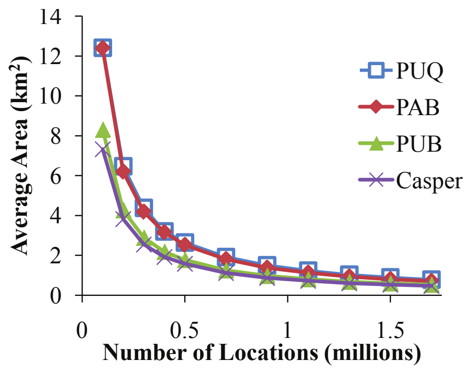
- Optimum policy-unaware quad tree (PUQ): since this was the first system [GG03] that proposed to use quad-tree based cloak to provide (policy-unaware) sender k-anonymity.

We could not use the original implementation of Casper in our experiments since the interface allows no bulk anonymization: Casper reads the input one location tuple at a time and for each location generates the cloak using only locations read up to that point. Instead of changing the original code, we decided to build a prototype of Casper based on the basic algorithm described in [MCA06]. We did not implement the adaptive algorithm since it only affects the running time and not the size of the cloak. We also implemented the policy-unaware quad-tree policy described in [GG03] that finds the smallest quadrant that contains the requesting location and at least k-1 other location as the cloak. We implemented the same approach over a binary tree to obtain an optimum policy-unaware binary tree.

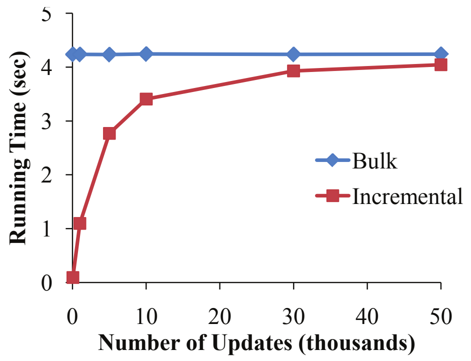
Figure 3.6(a) shows the comparison of average cloak areas obtained using the 4 algorithms described above. The horizontal axis represents the number of locations in the location database, and the vertical axis represents the average area (in square meters) of anonymized regions. k is fixed at 50. As expected, Casper has the minimum average cost among all the policies since it can select between horizontal or vertical semi-quadrants in contrast to fixed horizontal or vertical semi-quadrants selected by the policy-unaware binary tree. The cost of policy-aware sender k-anonymous policy is nearly identical to that of the policy-unaware quad-tree, and is at most 1.7 times that of Casper.

3.6.3 Effect of Optimizations

We evaluate the improvement of running time given by the optimizations presented in Section 4.4. We implemented our naive $Bulk_{dp}$ algorithm described in section 3.4.2 and then introduce the optimizations one at a time to obtain five different versions of our anonymizing algorithm. These includes $Bulk_{dp}$, $O(mn^3)$, $O(m(kh)^3)$, $O(m(kh)^2)$ and $O(m(kh)^2) + pruning$. In Figure 3.7 we show the time required by $O(mn^3)$, $O(m(kh)^3)$ and $O(m(kh)^2)$ to compute the optimal configurations for different size of location database and fixed $k = 50$. The x-axis shows the number of locations



(a) Average cloak area for various policies



(b) Incremental Maintenance time for $|D| = 1M, k = 50$

Figure 3.6: Parallel and Incremental Anonymization

in the location database and the y-axis shows the time in seconds. We decided not to include the running time for $Bulk_{dp}$ since it was significantly slower in comparison to $O(mn^3)$ even on small dataset of 1000 locations. The graph shows a big advantage in performance by $O(m(kh)^2)$ even on small inputs.

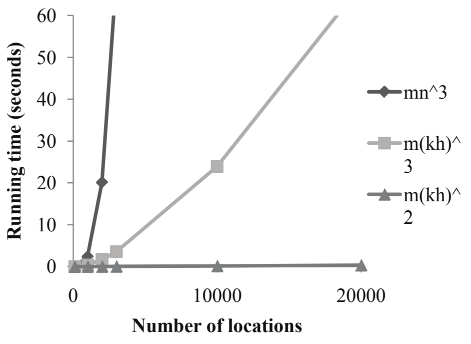


Figure 3.7: Effect of Optimizations on $Bulk_{dp}$

Figure 3.8 shows the comparison of time required by $O(m(kh)^2)$ and $O(m(kh)^2)$ + *pruning* for computing optimal configuration on much larger datasets and $k = 50$. The improvement in running time obtained due to pruning described in section 4.4 is more than 50%. In particular, the most optimized version manages to compute the optimal configuration for 1M locations in 8.5 seconds.

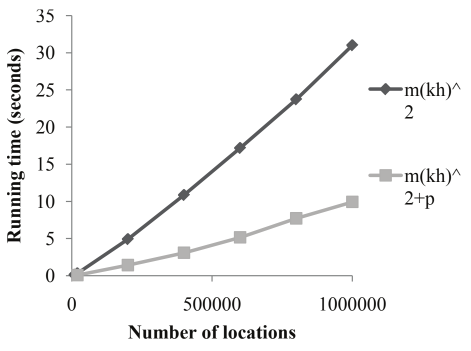


Figure 3.8: Effect of Pruning on Optimized $Bulk_{dp}$

3.6.4 Incremental Maintenance

We have also studied the performance of incrementally maintaining the optimum configuration matrix M from (the optimized version of) algorithm $Bulk_{dp}$. For the case of 1M locations and $k = 50$, we varied the number of locations that move from one snapshot of the location database to another. To this end, we randomly selected a set of distinct users updated their locations to a point at a randomly selected distance (bounded by 200 meters, that represents the maximum possible movement within 10 seconds) in a randomly selected direction. Figure 3.6(b) shows the comparison of performance of incremental maintenance with bulk re-computation. As expected, the time for incremental maintenance of M is always below that of the bulk re-computation as we increase the percent of moving users. However, we were surprised to notice that, once this percentage reaches 5%, the two times become virtually identical, and there is no gain in incremental maintenance. This is because most binary tree leaves require updating in that case, and incremental degenerates into bulk anonymization.

3.6.5 Utility Loss in Parallel Anonymization

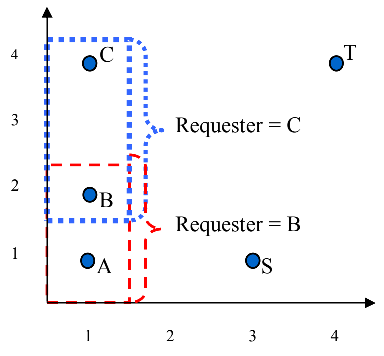
Recall from Section 3.5 that one concern when splitting the map into jurisdictions is the sub-optimal utility due to cases when the best cloak to anonymize a location is missed because it spans jurisdictions. As discussed in Section 3.5, we predict in general only a minimal divergence from the optimal cost. We verified this expectation experimentally by a stress test in which we increased the number of jurisdictions beyond reasonable limits: despite the above experiment showing that 16 suffice. Up to 2096 jurisdictions, the measured cost was identical to that of the optimal policy for the single-jurisdiction case, while the cost overhead for up to 4096 jurisdictions remained less than 1%.

3.7 Related Work

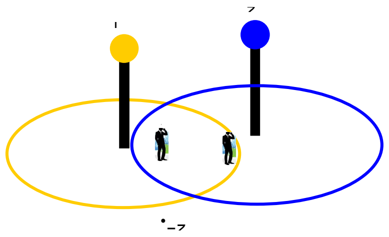
In the context of LBS, the two aspects of user privacy that have received the most attention are *location privacy* [BS03] and *sender anonymity*. The line of work on location privacy is complementary to this paper, as location privacy refers to hiding the precise location of the user (one is not required to hide the identity of the user) while sender anonymity refers to hiding the identity of the user (one is not required to hide the location, on the contrary, one assumes it falls in the attacker's hands). As described in the introduction, the extensions to *user-defined k* and *trajectory-aware attacker* are out of the scope of this paper and we leave them as future work.

Extensions of k-inside. Most of the proposals for sender anonymity are based on *k-anonymity* [Swe02b]. While a majority of these [GG03, MCA06, XC07] are simply based on the *k-inside* policy (described earlier, and shown to not defend against policy-aware attacks), some use variations. In [GL05], the cloaking policy ensures that at least $k - 1$ other users issue LBS requests from the cloaked region. It's been shown [GG03, KGMP07, CM07] that a k-inside policy fails to provide sender k-anonymity to "outlier" locations in some cases. To address this issue in k-inside policies, additional constraints of *k-reciprocity* [KGMP07] and *k-sharing* [CM07] have been proposed. *k-reciprocity* requires that among the $\geq k$ locations inside the cloak R of a location x, at least $k-1$ have x in *their* cloak, while *k-sharing* requires that at least $k-1$ of them have R as *their* cloak. We found that these additional constraints also fail to provide sender k-anonymity against a policy-aware attacker.

Consider the cloaking algorithm in [CM07] that takes into account the requesting locations to generate cloaking *groups* (set of locations that are cloaked to the same region). For locations in Figure 3.9(a), if the first request is made by C the algorithm groups C with B where as if the first request is made by B then it puts B and A in the same cloaking group to satisfy 2-sharing property. In the case when the initial request contains the cloak corresponding to $\{C, B\}$, a policy-aware attacker can infer that the sender is C!



(a) with 2-sharing



(b) with 2-reciprocity

Figure 3.9: Privacy breach

Next, consider a cloaking algorithm that generates circular regions centered at the base station nearest to the cloaked location. As shown in Figure 3.9(b) user Alice is closest to station S_1, hence her cloak is centered at S_1. User Bob is closest to station S_2 so his cloak is centered at S_2. Since both users are inside the intersection of both circular cloaks, this cloaking satisfies 2-reciprocity. When a policy-aware attacker observes the cloaking region centered at S_1, he can infer that the only possible sender is Alice!

Utility-maximizing cloaking. The problem of finding optimum k-anonymous cloaking that preserves privacy against a policy-unaware attacker has been considered (in [KGMP07]) to be NP-hard, by borrowing the results [MW04] from data k-anonymity. In [GL05] the authors study the problem of finding optimum cloaking using a minimum bounding box as the cloak and found it to be NP-hard and thus provide an approximate algorithm. Moreover it only uses the locations with pending requests for generating the cloak, as a result the cloak size can be quite large as not all users in a small region are expected to use LBS at nearly same time. The *FindMBC* algorithm in [XC07] computes the minimum bounding circular cloak that preserves privacy against a policy-unaware attacker. By Theorem 1, extending it to optimal policy-aware anonymization is likely hard.

Data k-anonymity. One may argue that some of the algorithms developed for data k-anonymization can be applied to the location database, to reduce sender k-anonymization to the classical data k-anonymization problem. Data anonymization algorithms come to mind that are based on *generalization* [Swe02a, LDR05, FWY05]. They require as input a generalization hierarchy for the anonymized data. One could conceptually use quad-tree based regions to represent such a generalization hierarchy for the location data. However, the reduction from sender to data k-anonymization is inadvisable since the problem of finding the minimum-cost data anonymization is known to be NP-complete [MW04], leading to algorithms for optimal anonymization that run in time h^n, where h is the height of the generalization hierarchy (quad-tree for us) and n is the number of tuples (the size of the location database for us). As we show here, more headway can be made by exploiting the additional structure of the problem, namely that the data to be anonymized is location data.

Some of the recent proposals [LW08, AFK$^+$06, BKBL07, CT07] for k-anonymizing data are based on clustering techniques. Most of these clustering algorithms [LW08, BKBL07, CT07] use a static distance metric where the cost of including a point x in a cluster C is independent of the cost of including another point y in C. For location data, since the cost model uses the area of anonymizing regions, the point farthest from the center of the cluster determines (and thus affects) the cost for other points in the cluster. This precludes a reduction from sender anonymity to cluster-based data anonymity. The cost model in a recent proposal [AFK$^+$06] takes care of this situation and thus its proposed clustering algorithm is a candidate to be investigated in future work for cloak generation in sender anonymity.

Private Information Retrieval. [GKK$^+$08] starts from the initial idea of having the requests mention no location information at all. The LBS sends all n points of interest *from the entire map* and the client filters them locally. An optimization consists in using cryptographic techniques that allows the sender to include its exact location in encrypted form, and the LBS to return $\sqrt{n}$ points of interest which include the ones closest to the sender. The result is directly obtained in encrypted form and its clear form is hidden even from the LBS.

This solution addresses a different point in the space of possible trade-offs of privacy versus feasibility. It achieves maximal anonymity since all senders are cloaked by the entire map area. The price to pay includes costly adoption, low throughput, and limited billing model. Adoption is hindered by the need to change the operation of current LBS to include cryptographic query evaluation. Throughput is impacted because cryptographic query evaluation is expensive: [GKK$^+$08] reports 20-45 seconds per query when the LBS maintains 65K points of interest. This time is lowered to 6-12 seconds per query when the computation is parallelized over 8 servers, depending on the length of the encryption key. Scaling to (tens of) thousands of requests per snapshot would therefore lead to either unacceptably slow response or prohibitively expensive massive parallelization. Secondly, the LBS's business model is limited because it does not know what query answers it returns. This precludes insertion of relevant ads, and rules out a Google-like model in which advertisers/service providers are charged by the volume of their ads/service postings reaching users. Since the LBS does not know what query answers it returns, this precludes insertion of relevant ads, and rules out a advertising-based business model in which advertisers/service providers are charged by the volume of their ads/service postings reaching users.

In contrast, our solution trades privacy (k is typically much lower than the number of all users) for feasibility: It requires virtually no change in existing LBS interfaces, whose input is essentially the one we describe above. It provides per snapshot a sub-second initialization time to bulk-anonymize over one million users, after which individual queries can be served in milliseconds. Indeed, serving a query requires looking up the location's cloak according to the computed policy, then evaluating a nearest-neighbor search for this cloak. Our experiments show that cloak lookup takes 0.3–0.5 ms. In [MCA06], Casper reports 2ms per nearest-neighbor query when $k = 200$ and there are 10K points of interest, using GIS indexing techniques. Adopting Casper's GIS-based query evaluation results in a per-snapshot throughput increase of 3 orders of magnitude over the cryptographic query evaluation of [GKK$^+$08]. Finally, flexible billing is facilitated since the LBS knows which advertisers/service providers to charge as it knows the query results.

Beyond k-anonymity: l-diversity and t-closeness. Taking a page from recent developments that improve on data k-anonymity, it is natural to ask if there are corresponding extensions of the notion of sender k-anonymity. The answer is positive, in the following sense. In data k-anonymity, there is a class of attacks based on counting the frequency of sensitive attribute values in the anonymized table. L-diversity [MGKV06] defends against the situation when all tuples in an anonymized group share the same sensitive attribute value, in which case they are all compromised. T-closeness [LLV07] goes beyond, defending even against attacks that compare the frequency of sensitive attribute values in the whole table against the frequency of sensitive attributes in individual anonymized groups. Whenever the two frequencies differ, the attacker learns something about the secret, and this fact is considered a privacy leak. The analogous attacks in our setting would consist of counting in each snapshot the number of duplicate requests, grouping by cloaking area and request values. For instance, the (unlikely) event of observing in a snapshot as many identical requests from the same cloak as the number of locations residing in it at that time exposes all senders. This assumes that a sender can issue a single request per snapshot, which is reasonable given the short snapshot duration.

The following simple modification of our approach to sender k-anonymity precludes the same class of frequency-based attacks as l-diversity and t-closeness: the anonymization server caches the query results returned by the LBS, indexed by the anonymized request. This means that the LBS does not even see duplicate anonymized requests during the same snapshot, and therefore cannot count their frequency (nor can it log it and thus make the count available to hacking or subpoena). For queries about stationary points of interest (most businesses and tourist attractions), the anonymizer can use the cache for a long time, thus precluding counting even across multiple snapshots. To adjust for the appearance and disappearance of points of interest, it suffices to flush the cache at infrequent intervals (for instance once a day). To help the LBS with billing, the anonymizer can keep a total count and submit it to the LBS at cache flushing time.

Anonymizer Architecture. The anonymizing framework in [GG03, GL05, MCA06, KGMP07, XC07] (and in this submission) is based on a centralized trusted anonymizer. The anonymizer has access to the location of all the users and their requests. We have not formally studied the impact of a distributed architecture of the anonymizer on our techniques. We have however shown empirically that, by running a farm of anonymization servers in parallel, we can arbitrarily reduce anonymization time while incurring less than 1% of cost overhead over the optimal cost.

Trajectories. Recent papers [BWJ05, XC07, CM07] have studied sender anonymity in the context of user "trajectories". The attacker has knowledge of when multiple requests have originated from the same (a priori unknown) user, even if they are sent at different times and from different locations. As it stands, our work is incomparable to [BWJ05, XC07, CM07], since we assume trajectory-unaware but policy-aware attackers, while [BWJ05, XC07] assumes trajectory-aware but policy-unaware attackers. We leave as future work the extension of the policy-aware anonymization algorithm presented here to handle trajectory-awareness.

3.8 Conclusion

We introduce the notion of sender k-anonymity against policy-aware attackers. This privacy guarantee is stronger than the sender k-anonymity in prior work, which defends against policy-unaware attackers only. Our results show that the novel guarantee strikes a pragmatic balance in the trade-off between strength of the privacy guarantee, utility, and running time for enforcement.

We also show the considerable amenability of the problem to parallelization, which reduces the anonymization time while preserving the optimal utility in virtually all cases. Indeed, dividing the San Francisco Bay area among 4K servers –far more than needed since 16 suffice– leads to only 1% divergence of the cost from the optimum. 16 servers already provide anonymization time of about half a second for 1 million users.

3.9 Proofs

3.9.1 Proposition 1

Proof. Since A provides policy-aware sender k-anonymity, there exists k PREs $\pi_1 \ldots \pi_k$ in $PRE(A, D, \mathbf{P})$ where $\mathbf{P} = \{P\}$. The policy P is a member of the family $\mathbf{P_C}$ of candidate policies that a policy-unaware attacker must consider. Therefore these k PREs also belong to $PRE(A, D, \mathbf{P_C})$ and witness that A provides sender k-anonymity against the policy-unaware attacker. □

3.9.2 Proposition 2

Proof. Without loss of generality lets assume that the k-inside policy picks cloaks of type *C* (where C could be any region type such as quadrants, circle or rectangle). The policy-unaware singleton-observer attacker only knows the type *C* of the cloaks. We model this by defining the set $\mathcal{P}$ to include all the policies that anonymize locations to cloaks of type *C*.

Let AR be an anonymized request observed by such an attacker. Let U be the set of locations in D covered by the cloak $R = reg(AR)$. The k-inside policy ensures that there are at least k locations in this cloak i.e. $|U| \geq k$. We randomly select k locations $l_1 \ldots l_k$ from U and construct k PREs $\pi_1 \ldots \pi_k$ such that for all $1 \leq i \leq k$, $\pi_i(AR) = SR_i$ where $loc(SR_i) = l_i$.

Thus a k-inside policy provides sender k-anonymity against the class of policy-unaware attackers. □

3.9.3 Theorem 1

Let D be an instance of location database, Γ be a set of possible centers and K and C are constants. Policy-aware Bulk-anonymization with circular cloaks decision problem, denoted as $Bulk(D, \Gamma, K, C)$, refers to finding a policy-aware sender k-anonymous policy P whose cost $Cost(P, D) \leq C$, where P uses circular cloaks each centered at some point from Γ with no restriction on the radius. For a given instance $Bulk(D, \Gamma, K, C)$, if a policy P k-anonymizes all the locations in D and its cost

$Cost(P, D) \leq C$, then we call it a *successful* policy. Next we show that finding successful policy for $Bulk(D, \Gamma, K, C)$ is NP-complete for cases $K \geq 4$.

Proof. **Polynomial time verification:** Let $Bulk(D, \Gamma, K, C)$ be an instance of a policy-aware bulk-anonymization with circular cloaks decision problem. Let P be a policy that uses circular cloaks centered at some point from Γ. We show now that the following can be checked in polynomial time:

- $Cost(P, D) \leq C$: For each user U in D, we create a service request SR (using arbitrary vector V of name-value pairs) and compute the cost of $AR = P(D, SR)$. The sum of costs of all the service requests is $Cost(P, D)$.
- *Policy-aware k-anonymous*: For every AR obtained above we can check if there are k distinct users ($u_1 \ldots u_k$) whose location is inside $reg(AR)$ and whose service requests ($SR_1 \ldots SR_k$) are anonymized to $AR_1 \ldots AR_k$ respectively under P and $reg(AR) = reg(AR_1) = \ldots = reg(AR_k)$.

Reduction: We reduce 3-satisfiability (3-SAT) [GJ90] to the Policy-aware Bulk-anonymization with Circular cloaks decision problem. Given an instance $S = (V, E)$ of 3-SAT problem, where V is the set of variables and E is the set of clauses, we create an instance $(D, \Gamma, 4, \pi \times |D|)$ of Policy-aware Bulk-anonymization with Circular cloaks decision problem where D is an instance of location database and Γ is a set of possible centers and the goal is to find a policy P that is policy-aware sender 4-anonymous such that $Cost(P, D) \leq \pi \times |D|$. We show that E in S is satisfiable *if and only if* we can find a policy P with above mentioned properties. Our reduction is inspired by [MS84] and we contrast our approach to them in the end. For ease of presentation we assume that a policy anonymize a location to a region.

We first describe the reduction at a high level. Let $S(V, E)$ be an instance of 3-SAT, to construct a corresponding instance of *Policy-aware Bulk-anonymization with Circular cloaks decision problem*, 3 special structures are used. Each variable $v_i \in V = \{v_1, \cdots, v_n\}$ has a corresponding "circuit of circles" (similar to one shown in Figure 3.11 that contributes locations to D and possible centers to Γ. Each clause $E_i \in E =$

$\{E_1, \cdots, E_m\}$ has a corresponding "Clause configuration" (as shown in dark dashed circles in Figure 3.12) that contributes locations to D and possible centers to Γ. A "clause configuration" determines how the circuits corresponding to the variables in the clause, relate to each other. It may not always be possible to draw the "circuit of circles" and "clause configurations" in euclidean space without intersecting circuits. To represent these intersections between circuits, a special structure called "Junction Configuration" is used. We call the geometrical representation of the constructed problem, a *circuit mash*. The locations introduced by these structures in the circuit mash has the following key properties (additional properties are presented later):

Property P1: For a circular region centered at one of the points in Γ to anonymize ≥ 4 locations, needs to have a radius $r \geq 1$.

Property P2: Each user location (along with 3 other user locations) can be anonymized to a circular region with radius 1 and centered at one of the points in Γ.

As we describe the construction of circuit mash of $Bulk(D, \Gamma, 4, \pi \times |D|)$ using the structures introduced above, we will show that the locations and possible centers contributed by each structure preserves P_1 and P_2. In turn, these properties (together with other properties described later) ensure that, S is satisfiable *iff* there exists a successful policy P for $Bulk(D, \Gamma, 4, \pi \times |D|)$ that only uses anonymized regions with radius 1.

Circuit for Variables: For each variable v_i we build a circuit $C^i = \{C^i_1 \dots C^i_{r_i}\}$ of circles where each circle has radius 1 and r_i is even (the exact value of r_i is determined at the end of the construction) with centers $p^i_1 \dots p^i_{r_i}$. In general, the distance between centers of two adjacent circles is 2 ($dist(p^i_j, p^i_{j+1}) = 2$) and the distance between centers of first and last circles is also 2 ($dist(p^i_1, p^i_{r_i}) = 2$), but this property will not hold for some circles that participate in junctions. The adjacent circles (that do not participate in the junction) intersect at a single point (i.e. they touch each other). The point of contact q^i_j of two adjacent circles C^i_j and C^i_{j+1} that do not participate in the junction is called *circuit critical point*. We include all the circuit critical points of a circuit in the set Γ (the set of possible centers). We put 2 user locations at the center of each circle in the circuit (except for some of the circles at the junction as explained later). Figure 3.10 illustrates a segment of a circuit, where numbers in the brackets indicates the number of users at the location. To anonymize any 4 locations in the circuit using a circular region centered at a circuit critical point it must have a radius $r \geq 1$. Additionally, any location can be 4-anonymized (along with 3 other user location) to a circular region of radius 1 that is centered on one of the circuit critical points. Thus the locations introduced by the circuit satisfies property **P1** and **P2**.

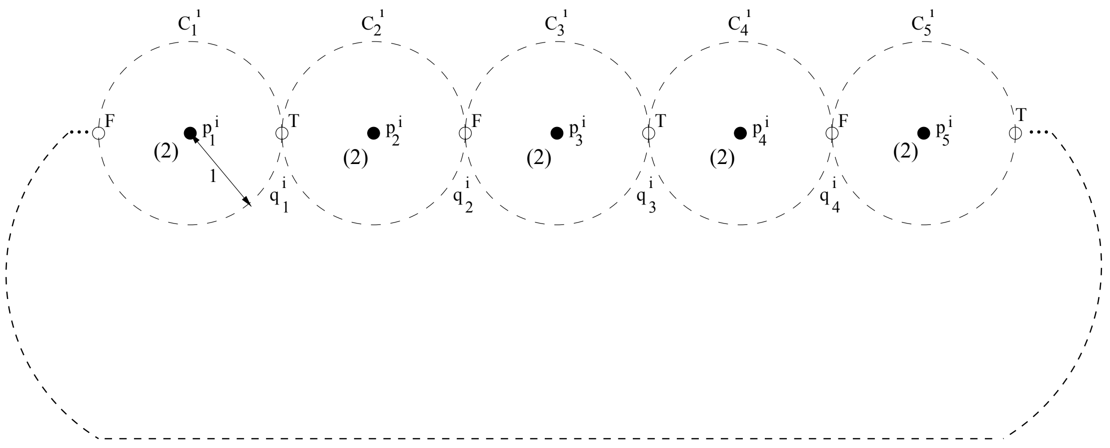


Figure 3.10: Locations of user tuples representing v_i

Next we describe a property that we later use to obtain a satisfying assignment for S, given a successful policy.

Property P3: In the circuit mash, for a circuit C^i corresponding to a variable v_i, if a successful policy uses a circuit critical point q_j^i as center of one of the anonymizing regions then it cannot use the circuit critical points q_{j-1}^i and q_{j+1}^i (where $2 \leq j \leq r_i$).

We prove this using contradiction. Suppose a policy P is successful and for a circuit C^i it uses two successive circuit critical points q_j^i and q_{j+1}^i as center of the anonymizing regions ar_j^i and ar_{j+1}^i respectively. Since P satisfies property **P1**, these regions must have radius 1 otherwise the total cost of anonymizing D will exceed $\pi \times |D|$. Since there are only 6 locations inside ar_j^i and ar_{j+1}^i, P cannot be policy-aware 4-anonymous and hence cannot be a successful policy, a contradiction.

In a circuit C^i corresponding to a variable v_i, we call the even numbered circuit critical points, q_j^i for j=0, 2, 4, $\cdots$ as *true* points. Similarily, the odd numbered circuit critical points q_{j+1}^i for j=0, 2, 4, $\cdots$ are called the *false* points. We can restate the property **P3**, as follows.

Property P4: In a successful policy, there are only two ways of chosing circuit critical points (as centers of anonymized regions) in a circuit: Either it chooses all the even points or all the odd points i.e. either it chooses q_j^i or q_{j+1}^i where j=0, 2, 4, $\cdots$.

As described later when creating a satisfiable assignment to S corresponding to a successful policy P, property **P4** is key to having a unique truth value assignment to the variable corresponding to the circuit.

In a successful policy, there are essentially two ways that circuit critical points can be chosen. Either it selects q_j^i or q_{j+1}^i where $j = 0, 2, 4 \ldots$. We call former the *true* circuit critical points and later the *false* circuit critical points. The first case can be visualized as assigning *true* value to v_i, and the latter case as assigning *false* value to v_i. From the point of view of cost it does not matter whether the anonymized regions are centered at true points or false points. Later when we describe the configuration for Clauses we will show how this choice affects the satisfiability of S.

Clause Configuration: Suppose we have a clause $E_s = (v_i \vee \neg v_j \vee v_k)$. In our reduction we use a configuration of 4 circles of radius 1 (that are not part of any circuit). The Figure 3.11 shows the dark dotted circles corresponding to E_s and how it is used to join the circuits corresponding to variables v_i, v_j and v_k. We draw 3 circles D_s^i, D_s^j, and D_s^k with radius 1 such that their centers Q_s^i, Q_s^j, Q_s^k form an equilateral triangle where length of each side is 2 i.e. each circle intersect the other 2 circles. The 3 points z_s^{ij}, z_s^{jk} and z_s^{ki}, where the circles intersect each other are called *clause* critical points and are included in the set Γ. We draw a 4th circle D_s^w with radius 1, centered at the circumcenter Q_s^w of the triangle formed by Q_s^i, Q_s^j and Q_s^k. We call Q_s^w the center of the clause configuration. Note that D^w includes z_s^{ij}, z_s^{jk} and z_s^{ki}.

We draw the circles D_s^i, D_s^j and D_s^k such that they pass through the circuit critical points of the circuits C_i, C_ji and C_k respectively. If the variable v_i in the clause is positive then we draw D_s^i such that it passes through a true circuit critical point of C^i and if the variable v_i in the clause is negative then we draw D_s^i such that it passes through a false circuit critical point of C^i. For each clause we select a circuit critical point that is not used by any other clause configuration. We place 1 user each at Q_s^i, Q_s^j and Q_s^k and 2 users at Q_s^w and include these users in D.

To 4-anonymize the 2 locations at Q_s^w, the circular region must have a radius 1 and centered at one of the z_s^{ij}, z_s^{jk} or z_s^{ki}. Without loss of generality let us assume the region is centered at z_s^{ij}. This region anonymizes 4 locations: the 2 locations at Q_s^w and 1 each at Q_s^i and Q_s^j. The user location at Q_s^k, cannot be anonymized by this region. To 4-anonymize the user location at Q_s^k, we must use a region of radius at least 1 that is centered at one of the circuit critical points in C^k. In particular if we use circuit critical point that intersects with circle D_s^k we can 4-anonymize location at Q_s^k with a region of radius 1. Clearly, the clause configuration satisfies both **P1** and **P2**.

Also if S is satisfiable then a successful policy satisfies **P3** even in the presence of clause configuration. In a clause configuration, there is only 1 location that can be anonymized to a region of radius 1 centered at the circuit critical point of a circuit. Therefore any successful policy P that selects two successive circuit critical points q_j^i and q_{j+1}^i as center of the anonymizing regions ar_j^i and ar_{j+1}^i (with radius 1) cannot be policy-aware 4-anonymous since there are only 7 locations (6 from the circuit and 1 from the clause) inside regions ar_j^i and ar_{j+1}^i.

We would like to draw reader's attention to the relation between anonymization of the 5^{th} location in a clause by a successful policy to a assignment for S. For a successful policy that satisfies **P3**, choice of the location that is anonymized using a region centered at circuit critical point also decides whether it chooses the true critical points or false critical points in a circuit. This in turn relates to the truth assignment for the variable corresponding to the circuit as explained later.

Junction: In order to make above configuration possible in euclidean space, we need to allow different circuits to intersect each other. Suppose the circuit C^i for a variable v_i need to intersect with circuit C^h for variable v_h. We design this intersection to make sure that property **P1** and **P2** are preserved and a successful policy satisfies **P3**. In order to do so we will describe the 2 exceptions to our circuit construction. We coincide a circle C^i_j (centered at p^i_j) of circuit C^i with circle C^h_k (centered at p^h_k) of circuit C^h such that they have a common center i.e. $p^i_j = p^h_k$ (we call this the *junction center*). Next we require that the points p^i_{j-1}, p^i_j and p^i_{j+1} are in a straight line which is perpendicular to a straight line joining the points p^h_{k-1}, p^h_k and p^h_{k+1}. Next we describe the 2 exceptions:

- Distance: We adjust the distances so that $d(p^i_{j-1}, p^i_j) = d(p^i_j, p^i_{j+1}) = d(p^h_{k-1}, p^h_k) = d(p^h_k, p^h_{k+1}) = \sqrt{2}$.
- Number of locations: We reduce the number of users located at each of the p^i_{j-1}, p^i_{j+1}, p^h_{k-1} and p^h_{k+1} to 1, and increase the users located at each of the p^i_{j-2}, p^i_{j+2}, p^h_{k-2} and p^h_{k+2} to 3. At the junction center ($p^i_j = p^h_k$), there are exactly two users (we remove 2 users).

Due to the reduced distance between the junction center and the centers of adjacent circles, the circle centered at the junction center now intersects with its following and preceding circles in the two circuits at 4 points TT, TF, FT and FF, We call them *junction critical centers* and add them to Γ, the set of possible centers. The junction critical centers also serves as the circuit critical (that no longer exists because of the reduced distance) in their corresponding circuits to satisfy property **P3**. The first alphabet in the name of a junction critical center indicates whether the point corresponds to a *true* or a *false* circuit critical center for the horizontal circuit and the second alphabet indicate the same for vertical circuit. Figure 3.12 illustrates the junction of the circuits C^i and C^h described above. In addition, we also require that there are at least 7 points between two successive junctions on a circuit.

This configuration preserves the property **P1** since to 4-anonymize the 2 locations at $p^i_j = p^h_k$, the anonymized region must have a radius at least 1 and centered at one of the circuit critical points. For other locations its easy to observe that an anonymizing region must have a radius of at least 1.

The junction also preserves the property **P2**. Any location at the center of the circles in the circuit (other than the junction center) can be 4-anonymized using one of the circuit critical centers. The 2 locations at the junction center can be 4-anonymized (along with 2 other locations at the center of adjacent circles) using a region of radius 1 that is centered at one of the junction critical centers.

Its not obvious that a successful policy can satisfy **P3** in the presence of a junction. We prove this using a counting argument. Without loss of generality, let P be a successful policy that selects two adjacent circuit critical centers q^i_{j-2} and q^i_{j-1} in the junction configuration as the centers of the anonymizing regions ar^i_{j-2} and ar^i_{j-1}. There are only 6 locations inside the two anonymizing regions and hence P cannot be policy-aware 4-anonymous. Even if one of the two adjacent centers is a junction critical point, (for e.g. q^i_{j-1} and TF in figure 3.12) there are still only 7 locations in the two anonymizing regions and therefore P cannot be 4-anonymous. Similarly if both the adjacent centers are junction critical points (for e.g. TF and FF in figure 3.12) there will be total 5 locations in the two anonymizing regions and therefore P cannot be policy-aware 4-anonymous.

IF: *If there is a satisfying assignment for 3-SAT then there is a policy-aware 4-anonymous policy P such that $Cost(P, D) \leq (\pi \times |D|)$.*

To show the IF part, suppose there is a satisfying assignment α to the instance S of 3-SAT problem. We construct a policy P_α that provides 4-anonymity using regions of radius 1 using α. Since all the regions in P_α are of radius 1, the cost $Cost(P_\alpha, D)$ = $|D|$ and thus P_α is a successful policy.

First we anonymize the locations introduced by the circuits of circles. Suppose C^h be a circuit of circles corresponding to variable v_h. If $\alpha(v_h)$ = *true* (or *false*) then we use all the *true* (or *false*) circuit critical points on the circuit C^h as centers and draw regions of radius 1. Each such region covers (and anonymizes) exactly 4 user locations introduced by the circuit. For e.g. a circle of radius 1 centered at q^h_i covers and anonymize 2 locations each at the centers p^h_i and p^h_{i+1} of adjacent circles C^h_i and C^h_{i+1}. Note that regions of radius 1 centered at all the *true* critical points anonymizes all the locations introduced by the circuit of circles. Same is true if we use all the *false* points as centers of regions.

Next we show how P_α anonymizes locations introduced by the clause configurations using the configuration of an example clause E_s shown in figure 3.11. Since α is a satisfying assignment, at least one literal of E_s must be *true*, w.l.g. lets assume it is v_i. If multiple literals are true we can use any one of them. In policy P_α, we anonymize the single user location at the Q^i_s to the region of radius 1 drawn at *true* circuit critical point of C^i that intersects with D^i_s. Note that in P_α this region is already used to anonymizes 4 locations of the circuit (since $\alpha(v_i)$ = *true*), therefore it can be used to anonymize location at Q^i_s. The 2 user locations at the center Q^w_s of the clause configuration along with 1 user location each at Q^j_s and Q^k_s are anonymized using a region of radius 1 centered at clause critical point z^{jk}_s. Note again that all the locations introduced by clause configurations are 4-anonymized using regions of radius 1.

We again use an example to demonstrate how P_α anonymizes locations at a junction. We consider the junction of two variables v_i and v_h as shown in Figure 3.12. Without loss of generality, let's assume $\alpha(v_i)$ = *true* and $\alpha(v_h)$ = *false*. We first draw a region with radius 1 centered at TF junction critical point. This region anonymizes the 2 locations at the junction center and 1 location each at p^i_{j-1} and p^h_{k-1} (the center of circles preceding the central circle). Note that if the truth assignment to (v_i, v_h) where different i.e. either (T, T) or (F, T) or (F, F) then we would have used region centered at TT or FT or FF respectively to anonymize the locations at the center of the junction. The location at p^i_{j+1} and the 3 locations at p^i_{j+2} are anonymized to a region of radius 1 centered at the true circuit critical point q^i_{j+1}. Note that this does not conflict with any choices made in the previous steps since $\alpha(v_i)$ = *true*. Similarly the 1 location at p^h_{k+1} and the 3 locations at p^h_{k+2} are anonymized to a region of radius 1 centered at false critical point q^h_{k+1}. Thus at a junction configuration, the policy P_α can anonymize all the locations using region of radius 1.

ONLY IF: *If there is a policy-aware 4-anonymous policy P such that $Cost(P, D) = (\pi \times |D|)$ then there is a satisfying assignment for 3-SAT problem S.*

In our reduction of S to $Bulk(D, \Gamma, 4, \pi \times |D|)$ we have shown that locations introduced by all our constructions satisfies property **P1**, and **P2** therefore there exists a successful policy in which all anonymizing regions are of radius 1. Let P be such a successful policy. We construct a satisfying assignment α_p for S as follows.

As described earlier, a successful policy that satisfies properties P_1 and P_2 also satisfies property P_3. This means that for any circuit C^i corresponding to a variable v_i, P either uses all the true points as centers of anonymizing regions or all the false circuit points. Thus if P uses all true points in C_i then we define $\alpha(v_i)$ = *true* and if P uses all false points then we define $\alpha(v_i)$ = *false*.

Next we show that α obtained from P as described above is a satisfies S. We show this using an example clause configuration shown in Figure 3.11. Since this configuration satisfies property **P1**, P must be using one of the clause critical points z_s^{ij}, z_s^{jk} or z_s^{ki} as center of an anonymizing region of radius 1. Let's assume it uses z_s^{ij} and anonymizes 2 locations Q_s^w and 1 each at Q_s^i and Q_s^j. The user location at Q_s^k, cannot be anonymized by this region. To 4-anonymize the user location at Q_s^k, P must use a region of radius 1 centered at true circuit critical point q_g^k. This means that P must have selected all the true critical points in the circuit C^k otherwise P cannot satisfy **P3**. Therefore $\alpha(v_k)$ = *true* and it would satisfy E_s. Similarly we can show that all the other clauses of E are satisfied by α and hence S is satisfiable. □

3.9.4 Lemma 1

Proof. **(a)** Let D be a location database with n locations and policies P_1 and P_2 are equivalent for anonymizing D w.r.t. quad tree T. We can describe the cost of anonymizing the n locations using P_1 as:

$$\begin{aligned} Cost(P_1, D) &= Cost(P_1(D, l_1)) + Cost(P_1(D, l_2)) + \ldots + Cost(P_1(D, l_n)) \\ &= area(m_1) + area(m_2) + \ldots + area(m_n) \end{aligned} \tag{3.1}$$

where $m_i \in T$ and m_i = $P_1(D, l_i)$ for $1 \leq i \leq n$. Note, for $i \neq j$, m_i and m_j can be the same node in T. Similarly we describe the cost of anonymizing the n location in D

using P_2 as:

$$\begin{aligned} Cost(P_2, D) &= Cost(P_2(D, l_1)) + Cost(P_2(D, l_2)) + \ldots + Cost(P_2(D, l_n)) \\ &= area(m'_1) + area(m'_2) + \ldots + area(m'_n) \end{aligned} \tag{3.2}$$

where $m'_i \in T$ and m'_i = $P_2(D, l_i)$ for $1 \leq i \leq n$. Note, for $i \neq j$, m'_i and m'_j can be the same node in T. Since P_1 and P_2 are equivalent, if a node $m \in T$ is used by P_1 to anonymize x locations then m is also used by P_2 to anonymize the same number of locations. Therefore

$$\begin{aligned} &area(m_1) + area(m_2) + \ldots + area(m_n) \\ &\qquad = area(m'_1) + area(m'_2) + \ldots + area(m'_n) \end{aligned} \tag{3.3}$$

because each quadrant appears same number of times on right sides of the Equation (3.1) and (3.2) for $Cost(P_1, D)$ and $Cost(P_2, D)$. Therefore,

$$Cost(P_1, D) = Cost(P_2, D)$$

(b) Let P_1 provides policy-aware sender k-anonymity to D. Therefore, each node $m \in T$, P_1 either anonymizes $(\geq k)$ locations using m or $none$. We are given that P_1 and P_2 are equivalent under D, T, therefore they both anonymize the same number of locations using m. Therefore, P_2 either anonymizes $(\geq k)$ locations using m or $none$. Thus, P_2 also provides policy-aware sender k-anonymity to D. Similarly we can show that if P_2 provides policy-aware sender k-aonymity to D, so does P_1.

Next we assume that P_1 does not provide policy-aware sender k-anonymity to D. Hence, there must exist a node $m \in T$ that is used by P_1 to anonymize $1 \leq i < k$ locations. Since P_1 and P_2 are equivalent, P_2 also anonymizes i locations using the node m. Since $1 \leq i < k$, P_2 also does not provide policy-aware sender k-anonymity to D. $\square$

3.9.5 Lemma 2

Proof. The cost of policy P for anonymizing the location database D with n locations as:

$$\begin{aligned} Cost(P, D) &= \sum_{l \in D} Cost(P(D, l)) \\ &= Cost(P(D, l_1)) + Cost(P(D, l_2)) + \ldots + Cost(P(D, l_n)) \\ &= area(m_1) + area(m_2) + \ldots + area(m_n) \end{aligned} \tag{3.4}$$

where $m_i \in T$ and m_i = $P(D, l_i)$ for $1 \leq i \leq n$. Since, for $i \neq j$, m_i and m_j can be the same node in T, we can rewrite the above equation as follows:

$$Cost(P, D) = \sum_{m \in T} f'(m, P) \times area(m)$$

where $f'(m, P)$ is the number of locations anonymized by P using quadrant m. Since C represents the equivalence class of P, the number of locations anonymized to each quadrant is same in P and C.

$$\forall m \in T, \quad f'(m, P) \times area(m) = f(m, C)$$

where $f(m, C)$ is as defined in Definition 8. Therefore the cost of configuration C on D can be written as:

$$\begin{aligned} Cost_c(C, D) &= \sum_{m \in T} f(m, C) \\ &= \sum_{m \in T} f'(m, P) \times area(m) \\ &= Cost(P, D) \end{aligned} \tag{3.5}$$

□

3.9.6 Lemma 3

Proof. Let P be a quad-tree policy and C be the configuration representing the class of policies equivalent to P.

First we assume that P provides policy-aware sender k-anonymity and show that C satisfies k-summation property. Since P is policy-aware sender k-anonymous, each quadrant $m \in T$ is used in P to anonymize either $\geq k$ locations or $none$. Thus

- for a leaf node $m \in T$
  - (i) If $d(m) < k$, then P cannot anonymize any location using m, therefore $C(m) = d(m)$.
  - (ii) if $d(m) \geq k$, then P could either anonymize $\geq k$ locations or $none$. In former case $C(m) \leq (d(m) - k)$ while in later case $C(m) = d(m)$.
- for an internal node $m \in T$ let $\Delta = \sum_{i=1}^{4} C(m_i)$, where $m_1 \ldots m_4$ are the children of m in T
  - (iii) if $\Delta < k$ then there are total ¡ k locations passed up by children of m. Thus P cannot anonymize any locations using m and therefore, $C(m) = \Delta$.
  - (iv) if $\Delta \geq k$ then the children of m passes up $\geq k$ locations. Therefore, P could either anonymize $\geq k$ locations or $none$. In former case $C(m) \leq (\Delta - k)$ while in later case $C(m) = \Delta$.

Thus C satisfies k-summation property. Moreover since P anonymizes every location to some region, $C(root) = 0$. Therefore, C is well-formed as well.

Next we assume that C satisfies k-summation property and show that P provides policy-aware sender k-anonymity. Equivalently we show that under C, each cloak of T is used to anonymize either $\geq k$ locations or $none$. Since C satisfies k-summation, it implies:

- for a leaf node $m \in T$
  - (i) if $d(m) < k$, then $C(m) = d(m)$. Thus P does not anonymize any location using m.
  - (ii) if $d(m) \geq k$, then either $C(m) = d(m)$ or $C(m) \leq (d(m) - k)$. In the later case P anonymizes $\geq k$ location using m, while in former case $none$.
- for an internal node $m \in T$ let $\Delta = \sum_{i=1}^{4} C(m_i)$, where $m_1 \ldots m_4$ are the children of m in T

(iii) if $\Delta < k$, then $C(m) = \Delta$. Thus P does not anonymize any location using m.

(iv) if $\Delta \geq k$, then either $C(m) = \Delta$ or $C(m) \leq (\Delta - k)$. In the later case P anonymizes $\geq k$ locations using m, while in former case $none$.

Thus P provides policy-aware sender k-anonymity. □

3.9.7 Lemma 4

Proof. We use structural induction to prove that each node m in the quad-tree T and an integer $l \leq d(m)$,

$$cost_{alg}(m, l) = cost_{min}(cset(m, l))$$

, where $cost_{alg}(m, l)$ represents the cost computed by $Bulk_{dp}$ for passing up l (unanonymized) locations at m and $cost_{min}(cset(m, l))$ represents minimum cost of passing up l locations at m among all the configurations that passes up l (unanonymized) locations at node m and satisfies k-summation property.

Basis: For a leaf node m and an integer $l \leq d(m)$, it is obvious by construction that $cost_{alg}(m, l) = cost_{min}(cset(m, l))$.

Induction: Let m be a non-leaf node in the quad-tree T and m_1, m_2, m_3, m_4 be the children of m. Let l be an integer such that $l \leq d(m)$ and $cset(m, l)$ be the set of configuration that passed up l (unanonymized) locations at node m and satisfies k-summation property. We show that $\forall g \in cset(m, l)$, $cost_{alg}(m, l) \leq cost(g(m))$, where $cost(g(m))$ represents the cost of g at node m. If $g(m_1) = l_1$, $g(m_2) = l_2$, $g(m_3) = l_3$, and $g(m_4) = l_4$, the cost of m in g can be written as

$$\begin{aligned} cost(g(m)) := [&cost(g(m_1)) + cost(g(m_2)) + cost(g(m_3)) + cost(g(m_4)) \\ &+ area(m) \times (l_1 + l_2 + l_3 + l_4 - l)] \end{aligned}$$

By induction hypothesis we assume that $cost_{alg}(m_1, l_1) \leq cost(g(m_1))$, and similarly $cost_{alg}(m_2, l_2) \leq cost(g(m_2))$, $cost_{alg}(m_3, l_3) \leq cost(g(m_3))$, and

$cost_{alg}(m_4, l_4) \leq cost(g(m_4))$. Therefore

$$\begin{aligned}&cost_{alg}(m_1, l_1) + cost_{alg}(m_2, l_2) + cost_{alg}(m_3, l_3) + cost_{alg}(m_4, l_4)\\ &\leq cost(g(m_1)) + cost(g(m_2)) + cost(g(m_3)) + cost(g(m_4))\end{aligned}$$

And by adding the constant value $area(m) \times (l_1 + l_2 + l_3 + l_4 - l)$ to both the sides we get

$$cost_{alg}(m, l) \leq cost(g(m))$$

Similarly for each node m and each integer $l \leq d(m)$, and each configuration $g \in cset(m, l)$, we can show that $cost_{alg}(m, l) \leq cost(g(m))$. Therefore $cost_{alg}(m, l) = cost_{min}(cset(m, l))$. □

3.9.8 Lemma 5

Proof. Let D be a location database instance, B a binary tree, C an optimal configuration of B, and P a policy it represents. Suppose there is a node $m \in B$ such that $(k+1)h(m) < C(m) < d(m)$. Then there is a set S of at least k locations such that (i) all locations in S occur in m's area, and (ii) all locations in S are cloaked by P using some ancestor a of m, and (iii) if all locations in S are removed from their respective cloaks under P, the cloaks continue to contain at least k locations. Construct a policy P' that cloaks the locations in S using m instead of its ancestors. P' continues to be policy-aware sender k-anonymous, but has lower cost, contradicting the optimality of P. □

3.10 Acknowledgements

My co-authors and collaborators, Kevin Keliang Zhao at UCSD and Richard Hull at IBM, together with Alin Deutsch, made important contributions to the material that formed the basis of this chapter.

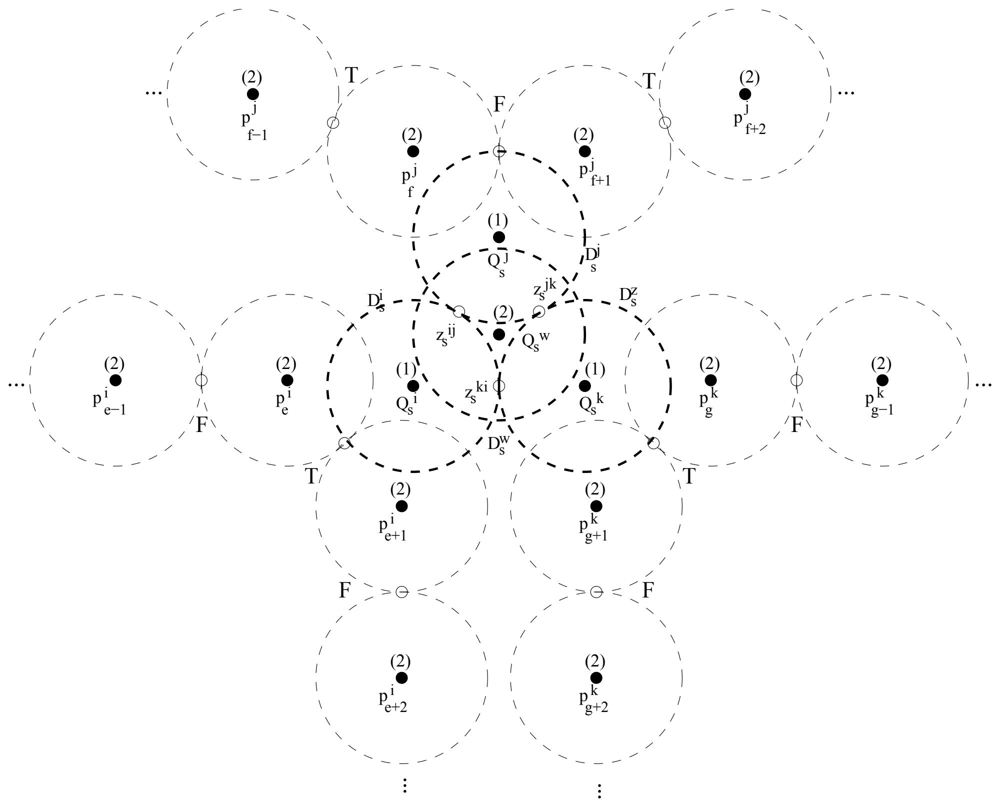


Figure 3.11: Representation of a clause

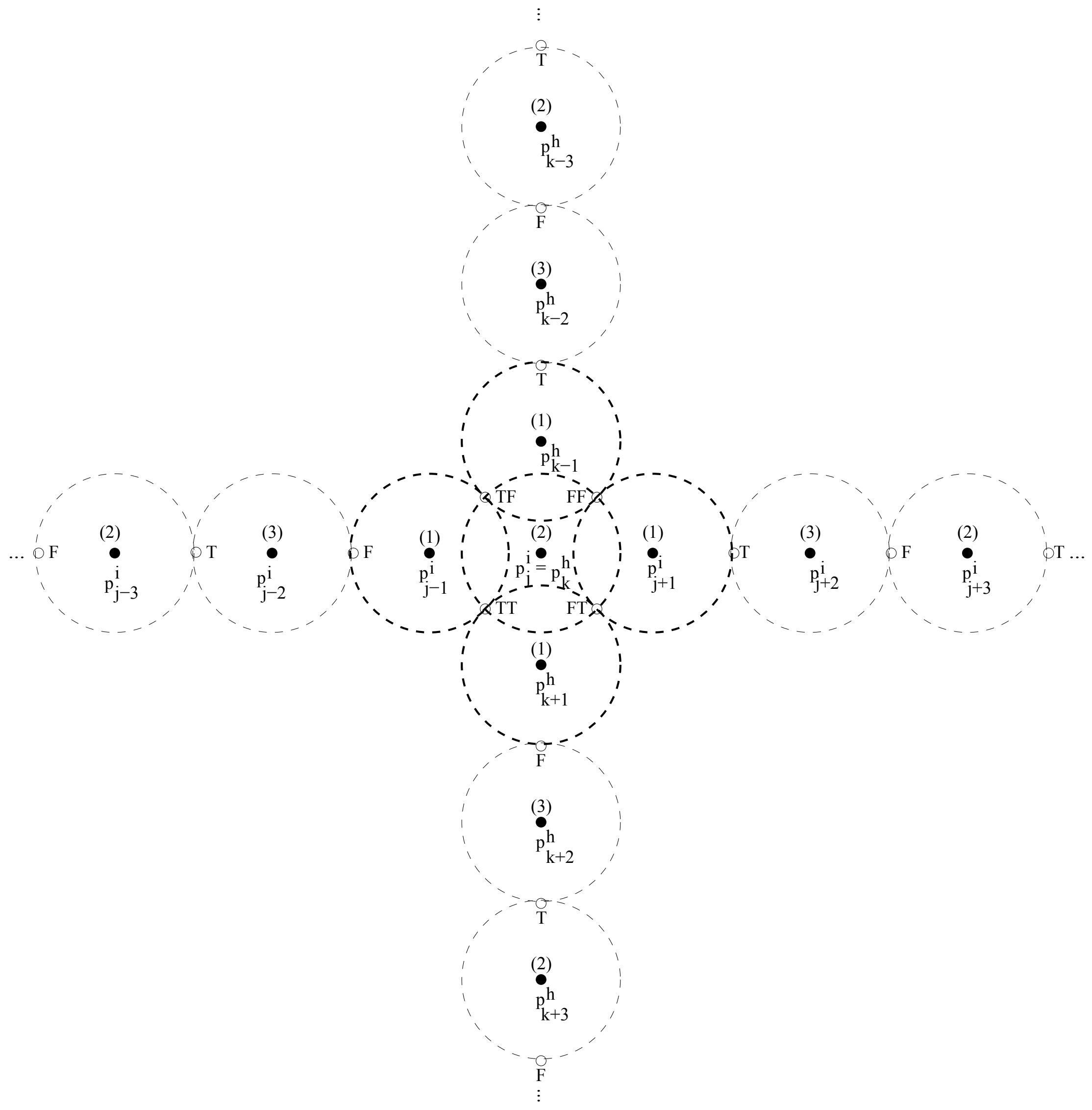


Figure 3.12: Junction configuration

Chapter 4

Trajectory-aware and Policy-aware Sender k-anonymity

4.1 Introduction

A Location-based service (LBS)[lbs] is an information or entertainment service, accessible with mobile devices through the mobile network and utilizing the geographic location of the mobile device (e.g. “find the nearest gas station”, “Thai restaurant”, “hospital”). In recent times, the availability and usage of Location-based service has increased significantly because the location information of mobile device can be computed automatically (without any input from the user) by the wireless network (via triangulation of mobile device signal) and at the mobile devices (via embedded GPS chipset).

Most of the popular Location-based services such as *Facebook Places* [fac], *FourSquare* [fou], *Gowalla* [gow] and *Loopt* [loo] log the LBS requests sent by their users. The data retention policies of these LBSs have provisions that describe this intent. We refer to these logs as *LBS request logs*. A LBS request log, containing LBS requests collected over a period of time, is of great value to the advertisers and researchers as it can be used to answer queries such as “the requests sent by users that move from a location *A* to location *B*” or “requests sent by the same user over a period of time”. But a LBS request log may also contain some sensitive requests of the users (e.g. for the local campaign headquarter of a given political party, spiritual center for a given religion). In the event an attacker gains access to a LBS request log, the privacy of the senders, who wants to keep their interests private, is at risk.

In this paper we investigate how to protect identity of the sender of the LBS requests against attackers who (via hacking or subpoenas or financial agreement) gain access to *a)* the LBS request log (from the LBS provider) and *b)* to the sequence of locations (*trajectory*) visited by the mobile users, for the duration the LBS requests are logged (from the wireless service provider or location computing servers such as Sky-Hook [sky]) and *c)* who know the "design" of the system used to provide this protection. The assumption *c)* is based on a well accepted principle of designing a private and secure system - "The design is not a secret" [Sal74]. The assumption *b)* is a realization of the fact that an attacker can obtain the locations visited by the users from many sources including surveillance. A recent article in *Wall Street Journal* [wsj] and a joint study [EGC$^+$10] by *Intel Labs*, *Penn State*, and *Duke* University provides the evidence that the advertisers are logging the trajectories of the mobile device users.

In the context of LBS, the best-studied identity protection measure is known as *sender k-anonymity* [KGMP07], which is intended to guarantee that the content of a LBS request and precise location of the users are insufficient to distinguish among the actual sender and k-1 other possible senders. This privacy guarantee is targeted towards the LBS requests sent by the users at a given instant of time. The underlying model does not consider the LBS requests sent by the users in the past (other instants of time). We refer to this privacy guarantee as *snapshot sender k-anonymity*, and any solution that targets such guarantee as *snapshot k-anonymization.*

Typical snapshot anonymization algorithms such as [GG03, MCA06, KGMP07, DHVZ10] are based on hiding the sender's precise location in the request, substituting instead a $cloak$, i.e. a region containing this location. The cloak is usually chosen from among regions of a pre-defined shape (circular, rectangular etc.), to include locations of at least k-1 other mobile users. The policy to choose the cloak takes into account the locations of the nearby users and the cloak selected for anonymizing any possible requests from them. We refer to such a cloak selection policy as *snapshot k-anonymous policy*. Next we show an example of a snapshot 2-anonymous policy.

Example 11. Figure 4.1 and Figure 4.2 show the location of five users at two instants

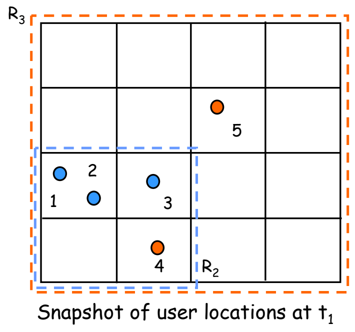


Figure 4.1: User locations at instant t_1

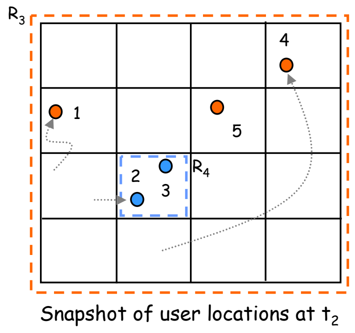


Figure 4.2: User locations at instant t_2

Table 4.1: Snapshot policy-aware 2-anonymous policies

P_1	P_2
...	...
$1 \rightarrow R_2$	$1 \rightarrow R_3$
$2 \rightarrow R_2$	$2 \rightarrow R_4$
$3 \rightarrow R_2$	$3 \rightarrow R_4$
$4 \rightarrow R_3$	$4 \rightarrow R_3$
$5 \rightarrow R_3$	$5 \rightarrow R_3$
...	...

t_1 and t_2. Table 4.1 shows the cloak selection policies P_1 and P_2 that use the quadrants, of a static quad-tree based partitioning of the geographic space, as cloaks. Suppose at instant t_1, user *1* sends a LBS request L_1. Policy P_1 anonymizes L_1 by substituting the location in the request with the cloak R_2 (shown in Figure 4.1). Note that R_2 includes the location of users 1, 2, 3 and a request sent by any one them is anonymized by P_1 using the same cloak R_2. Thus when an attacker, who has access to the user locations at

t_1 and policy P_1, observes the anonymized request with cloak R_2, he cannot distinguish whether the sender is user 1 or 2 or 3. Thus P_1 provides snapshot sender 2-anonymity.

Suppose user *1* sends another LBS request L_2 at instant t_2. Policy P_2 anonymizes L_2 by substituting the locations in the request with cloak R_3 (shown in Figure 4.2). An attacker who has access to user locations at t_2 and policy P_2 cannot distinguish the sender among users 1, 4 and 5 since requests from all these users are anonymized using R_3. Thus P_2 provides snapshot sender 2-anonymity. Note that policy P_2 does not take into account the user locations at instant t_1 and their anonymizations using policy P_1 (and vice versa).

Thus an obvious approach to preserve sender k-anonymity of users, whose LBS requests are logged, is to anonymize the LBS request logs using one of the snapshot k-anonymization solutions [GG03, MCA06, KGMP07, DHVZ10]. Our next example demonstrates the anonymization of a LBS request log using the snapshot k-anonymization policy of Example 11.

Example 12. Consider the locations of five users, at two instants t_1 and t_2, shown in Figure 4.1 and Figure 2. User 1 sends LBS requests L_1 (at t_1) and L_2 (at t_2), which are logged by the LBS. The LBS uses the snapshot anonymization policies P_1 and P_2 (shown in Table3.1) to 2-anonymize L_1 and L_2 respectively, as described in Example 11. To preserve the linkage information that both the requests were sent along the same trajectory, the LBS inserts same pseudo-id in both the anonymized requests.

Unfortunately snapshot anonymization of a request log does not provide sender k-anonymity against an attacker who has access to the LBS request log and the locations of all the users for the period the requests are logged. This is because, while anonymizing LBS requests in a given instant, a snapshot sender k-anonymous policy does not take into account the anonymization of LBS requests in other instants. The next example describes a scenario when sender k-anonymity can be breached if the LBS request log is anonymized using a snapshot k-anonymous policy.

Example 13. Lets consider the snapshot k-anonymization of request logs of Example 12. Lets assume the anonymized request log is observed by an attacker who knows the policies P_1 and P_2 and the locations of users in the two instants the request were sent (shown in Figure 4.1 and Figure 4.2. As described in Example 11, the attacker can use the knowledge of the policies to identify the sender for the first request to be one of the 1, 2 or 3 and that for the second request to be one of the 1, 4 or 5.

Next he observes that both the requests where sent by the same user. Since user 1 is the only common user across the set of possible senders for the two requests, the attacker successfully concludes that user 1 must be the sender, breaching sender 2-anonymity! □

In Example 13 the attacker is able to breach sender 2-anonymity because the request logs enables him to associate the two requests to the same trajectory and the snapshot 2-anonymization policy P_2 does not take into account the anonymization of request from user 1 at instant t_1 using policy P_1. Thus the above breach could have been avoided if either the two requests were not linked with the same trajectory or if instead of policy P_2 we use a policy P_2' that anonymizes requests from users 1, 2 and 3 using the region R_3. While we are independent to change the anonymization policy to preserve sender k-anonymity, we do not wish to entirely remove the association of requests with a trajectory in the anonymized LBS request log since this is a valuable information. This poses an interesting challenge for the LBS: how can it publish the LBS request logs and some form of linkage information between requests and the trajectories, without jeopardizing k-anonymity of the users? The LBS needs to ensure this against an attacker who knows locations of all the users for the period the LBS requests are logged and who knows the "design" i.e. the policy used to pick cloaks for anonymizing requests. We call this problem the *offline TP*-aware sender k-anonymity problem since a LBS request, sent by a user in the past, can be anonymized after observing his future requests. We contrast this with the problem of *online TP*-aware sender k-anonymity, in which

- The LBS is not trusted, therefore a LBS request is anonymized before it is sent to the LBS.
- the anonymization of a LBS request, sent by a user in the past, cannot be changed or altered after observing his future requests.

We leave the problem of online *TP*-aware sender k-anonymity for future work.

In this paper, we propose a solution to the offline trajectory-aware sender k-anonymity problem. Instead of anonymizing each of the LBS requests sent by a user independently, we anonymize all the requests from the same user together. To this effect we propose to use a sequence of cloaks to anonymize the set of LBS requests sent by a user, over a period of time. With each cloak in the sequence we associate a set of LBS requests devoid of any location and sender identity information. Such a sequence S of cloaks represents LBS requests sent along a *set* of user trajectories that pass through S, thus preserving the coarser association between LBS requests and the trajectories they were sent along. Moreover S provides offline trajectory-aware sender k-anonymity, if it represents the LBS requests of k distinct users whose trajectories passes through the sequence of cloaks. We describe our approach in detail in Section 4.3.

Our contributions. In addition to showing that snapshot k-anonymization policies, if used for anonymizing LBS request logs, can jeopardize sender k-anonymity against an attacker who has complete knowledge of the snapshot k-anonymization policy and the trajectories of the users, our contributions include the following.

[1] We identify and formulate the problem of offline sender k-anonymity in LBS request logs against the class of trajectory-aware and policy-aware attackers. We extend the snapshot sender k-anonymity guarantee and define a novel *TP*-aware sender k-anonymity guarantee.

[2] We study the problem of finding, among all the offline policies that provide *TP*-aware sender k-anonymity, one with the optimum utility. We show that finding the optimum offline policy that uses cloaks that are chosen among the quadrants of a quad-tree based partition of the map is *NP-Hard*.This is significant since it was shown in [DHVZ10] that for such cloak types optimum snapshot k-anonymization is P-time.

[3] We show that the problem of optimum *TP*-aware sender k-anonymity is approximable (i.e. one can always find a solution that is within well defined bounds relative to the optimum solution).

[4] In particular, we describe a novel l-approximation algorithm to anonymize a LBS request log spanning user trajectories of length l and novel optimizations to improve the average cost of anonymization.

[5] We implement and experimentally evaluate our offline trajectory-aware sender k-anonymous algorithm and show that it is practical and scales extremely well with the number of user trajectories: it takes less than 5 minutes to anonymize 1 million user trajectories of length 30 in the San Francisco Bay area.

Paper outline. The remainder of the paper is organized as follows. In Section 4.2, we describe a prevailing model of an LBS and the logs maintained by the various entities involved in the delivery of the LBS. We define offline trajectory-aware sender k-anonymity in Section 4.3 and describe our solution that uses a sequence of cloak to preserve trajectory-aware sender k-anonymity while publishing blurred linkage information between anonymized requests and set of user trajectories. In Section 4.4 we show that finding the optimum offline policy that provides trajectory-aware k-anonymity is NP-hard and hence in Section 4.4.2 we propose a polynomial time l-approximation algorithm. In Section 4.5 we describe the optimizations and our implementation of optimized approximation algorithm. We report on the experimental evaluation in Section 4.6, discuss related work in Section 4.7 and conclude in Section 4.8.

4.2 Location Based Services

This section introduces a prevailing model of location-based services, based on automatic computation of the location of user mobile devices. It describes various entities in the LBS ecosystem, the data flow among these entities and the data logged by them.

4.2.1 Basic LBS Model

As shown in Figure 4.3 there are four core elements in the delivery of a location-based service: the user making a request, typically called the *sender*, the (wireless) Communication Service Provider, denoted as *CSP*, the location server that computes the location of the mobile device, denoted as *LS*, and the Location Based Service (LBS) provider, denoted as *LBS*. We view the CSP, Location Server and LBS provider to be trusted agents and assume that the communication between them is secure.

To access an LBS, a sender uses an application on the mobile device (typically provided by the LBS). The application fetches the location from the run-time environment on the mobile device, which in turn gets it from the location server. The location

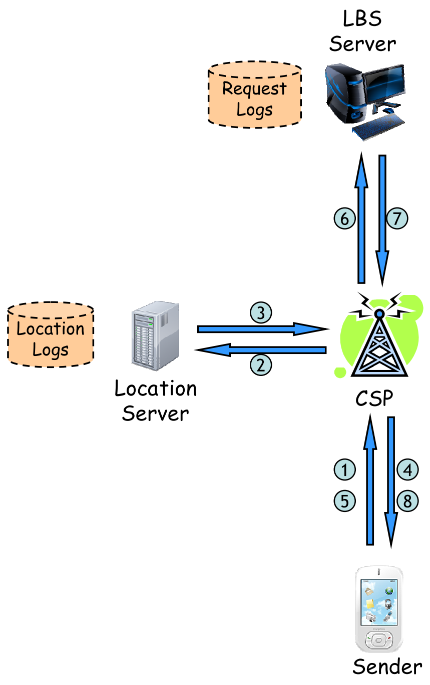


Figure 4.3: LBS Model

server is a specialized network component in CSP's network, known as *Mobile Positioning Center* (MPC) in the CDMA standard, that provides access to device locations for E911 [E91] and other location-based services. The location can also be obtained from a service that operates outside the CSPs network and can compute the location of a mobile device using the signal strength of nearby cell-towers and WiFi access points observed on the mobile device (e.g. SkyHook [sky] and Google Location Service [gls]). The application then sends a service request containing the location and the specifics of information/operation requested by the sender (e.g. "car dealership in 5 mile radius from my location" or "notify me when a friend is within 1 mile from my location". The LBS provider responds to the LBS request using the location sent with the request.

Henceforth we abstract from these details and focus on the treatment of location data and the LBS service requests in the LBS ecosystem. For simplicity of presentation, we model a geographic area as a 2-dimensional space and user's location as integer coordinates within this 2-dimensional space.

Location Server. The Location Server (LS) is a logically centralized point that provides access to the locations of mobile devices. We abstract from the fact that location is usually determined only on demand, and assume in our investigation that the locations of all devices are eagerly computed and available. We also assume that the location server logs the location of all devices after every fixed interval of time. While the actual duration of the interval is not important to our discussion, we do assume that the LS logs the locations of all the users at the *same* instant in time, thus capturing a “snapshot” of user locations at that instant. We refer to the sequence of the snapshots of user locations as the *location log* and the locations of a user in the sequence of snapshots represent his trajectory.

For ease of presentation, we use consecutive natural numbers to represent the time instant for consecutive snapshots, starting with ’0’ for the first snapshot. Note that it still preserves the ordering of locations visited by the users. Our techniques are equally applicable to currently prevailing representations for time (e.g ISO 8601 [tim]).

In this paper we only consider locations logs with finite number of snapshots, and consequently, finite number of locations in a user trajectory. The number of snapshots in a request log is referred to as the length of the request log and the number locations in a trajectory is referred to as the length of the trajectory. Table 4.2 represents a location log containing five user trajectories of length 2.

Table 4.2: Location log of length two

userid	1	2
Alice	(1,2)	(1,3)
Bob	(1,2)	(2,2)
Carol	(2,2)	(2,2)
Sam	(2,1)	(4,4)
Tom	(3,3)	(3,3)

Our assumptions about location logs are motivated by applications such as *SpotRank* [spo]. SpotRank claims to correctly predict the number of users at a given location, at a given day and time. To make the prediction, it uses the location logs collected by the Skyhook location server [sky] over a period of time. To enable such a service, the location logs must be organized to find the location of all the users at the same instant of time.

LBS Provider. The LBS provider uses the location in the LBS requests to respond to the user query or perform the requested operation. We assume that after responding to each LBS request, the LBS provider logs the request. We refer to this log as the *LBS request log*. Each logged request is associated with the identifier of the device that sent the request (e.g. IP-address or MAC-address). This allows the LBS provider to identify the requests sent by the same user in the LBS request log. This assumption reflects the prevailing data retention and privacy policies among the popular LBSs such as *Facebook Places*, *FourSquare*, *Loopt* and *Gowalla*.

Table 4.3: LBS request log

deviceid	1	2
id_a	v_1	v_2
id_b	v_3	-
id_c	-	v_4
id_s	v_5	-
id_t	-	v_6

Example 14. Let us consider the five users whose trajectories are shown in Table 4.2. Suppose at instant 1, a LBS L logs the requests v_1, v_3 and v_5 sent by the users Alice, Bob and Sam (respectively). At instant 2, L logs the requests v_2, v_4 and v_6 sent by the users Alice, Caroll and Tom (respectively). Table 4.3 shows the request log of L where id_a, id_b,id_c,id_s and id_t are the device-ID of Alice, Bob, Caroll, Sam and Tom respectively. $\square$

Our assumptions about the location server and the LBS provider are not only indicative of the current prevailing data collection practices, but are also appropriate in connection with the study of privacy guarantees. This is because we target attackers who might be able to reconstitute all device locations, perhaps by hacking logs, or by financial agreement or by subpoena-induced cooperation of the location server and the LBS provider.

As a mobile user moves and sends LBS requests from different locations at different times, the LBS provider logs these requests and using the device-id builds a *history* of LBS requests sent by the same user. Although this history does not contain the identity of the sender, it can be linked with the user trajectories in the location log at the location server and thus can be associated with the actual sender. Therefore we need way represent all all the data corresponding to a user that is available in the LBS model described above.

4.2.2 User History

Next we define an abstract class *User-history* that represents the data corresponding to a LBS user, stored in the location log and the LBS request log. The class user-history consists of two core methods, $location()$ and $request()$, defined over the set of natural numbers. The method $location()$ maps the natural numbers to a set of points in the 2-dimension coordinate space and the function $request()$ maps the natural numbers to LBS requests in the LBS request log.

Definition 10. The user-history is an abstract class with following three methods:

i) **rid()** returns a id, where domain of id is the set of all mobile device identifier of users. (e.g. IP-address or MAC-address).
ii) **location(i)** takes as an input a natural number i and returns a point (x_i, y_i) in the 2-dimensional integer coordinate space (used to represent a geographic area).
iii) **request(i)** takes as an input a natural number i and returns the set of name-value pairs $\{V_1 \cdots V_q\}$ that represents a set of LBS requests, each request devoid of any location information.

In the above definition, the set of name-value pairs $\{V_1 \cdots V_q\}$ contain the categories and specifics of the sought services (e.g. $[(poi, rest), (cat, ital), (dist, 2mi)]$ represents “find all the Italian restaurants within 2 miles of my location”). It preserves the ordering of locations in the user trajectory as witnessed in the location logs. It also preserves our snapshot assumption (described above) about the location log since for any natural number i, and for any two users u and v, the locations $u.location(i)$ and $v.location(i)$ represent their location in the *same* instant in time i.e. same snapshot .

At a high level, the function $location()$ represents the trajectory of a user and the function $request()$ represents the set of LBS requests sent by that user along his trajectory. In particular, $u.location(i)$ is the location of the user $u.userid()$ as observed in the snapshot i of the location log and $u.request(i)$ is the set of LBS requests in the LBS request log that were sent from $u.userid()$ from the location $u.location(i)$.

Since in this paper we focus on user trajectories of finite lengths, we say a a user history is of length l if the functions $location(i)$ and $request(i)$ are defined only for $1 \leq i \leq l$. Henceforth, we represent a user-history of length l as

$$(uid, \langle loc_1, loc_2, \ldots loc_l \rangle, \langle V_1, V_2, \ldots V_i \rangle)$$

where uid is the userid of the user, loc_i is the location of the user at instant i and V_i the request sent by the user at instant i from l_i. In the rest of the paper, we use the term user-history to refer to an object of the user-history class.

Example 15. Consider the location log shown in Table 4.2 and the LBS request log shown in Table 4.3. The user-history corresponding to the five users whose data is in these logs are as follows:

u_a = (Alice, $\langle (1,2), (1,3) \rangle$, $\langle v_1, v_2 \rangle$)
u_b = (Bob, $\langle (1,2), (2,2) \rangle$, $\langle v_3, \rangle$)
u_c = (Carrol, $\langle (2,2), (2,2) \rangle$, $\langle , v_4 \rangle$)
u_s = (Sam, $\langle (2,1), (4,4) \rangle$, $\langle v_5, \rangle$)
u_t = (Tom, $\langle (3,3), (3,3) \rangle$, $\langle , v_6 \rangle$) □

Even though a user-history is not materialized and available as shown above, it is a succinct and accurate representation of information about a LBS user that can be obtained by an attacker (as described earlier) in our LBS model. Thus the data available in the LBS model (in the logs), corresponding to a set of LBS users, can be represented as a set of user-history objects. The set of user-history objects contains very useful data for researchers and advertisers since it can be used to answer queries such as "the requests sent by users that move from a location *A* to location *B*". On the other hand it also contains some sensitive requests of the users and other private information such as as what LBS requests were sent by a user along a particular trajectory.

This brings us to the privacy issue that we study in this paper i.e. how can the LBS provider anonymize the set of user history objects such that it preserves some form of linkage between the service requests and the trajectories, but an attacker who has access to the anonymized user history objects and who knows the policy used for anonymization, cannot breach sender k-anonymity of the users. As shown in the Example 13 in section 3.1, the snapshot sender k-anonymity solutions cannot provide sender k-anonymity in this scenario.

4.3 TP-aware k-Anonymity

In this section we describe our approach for anonymizing the set of user-history objects to provide sender k-anonymity against the class of attackers who are aware of user trajectories and the anonymization algorithm. The anonymization preserves the linkage information between LBS requests and trajectories to an extent that does not pose any risk to the sender k-anonymity of the users.

4.3.1 Bundles

Our approach is based on the observation made in section 3.1 that a LBS request in the LBS request log should not be anonymized without considering the anonymization of other LBS requests in the LBS request log sent along the trajectory of that user. To overcome this problem, in our approach we anonymize all the requests sent along a trajectory of a finite length l, simultaneously, using a sequence of cloaks.

To this effect, we define an abstract class *bundle* that consists of two core methods, $cloak()$ and $requests()$, defined over the set of natural numbers. The function $cloak()$ maps the natural numbers to 2-dimension cloaks of a pre-determined shape (e.g. circular, rectangular or quadrants of a quad-tree) and the function $requests()$ maps the natural numbers to sets of LBS requests.

Definition 11. [**Bundle**] A bundle object **b** is an instantiation of an abstract class with following three methods:

i **id()** returns an identifier (e.g. integer) that uniquely identifies the bundle.

ii **cloak(i)** takes as an input a natural number i and returns a 2-dimensional region c_i.

iii **requests(i)**takes as an input a natural number i and returns the set of name-value pairs $\{V_1 \cdots V_q\}$ representing a set of LBS requests, each request devoid of any location information.

In the above definition c_i is a finite representation of a 2-dimensional region (e.g. $[(x_1, y_1)\ (x_2, y_2)]$ for axis-parallel rectangles, where (x_1, y_1) and (x_2, y_2) are coordinates of lower-left and upper-right corners of rectangle respectively). We denote a bundle as a tuple = $[rid, \langle r_1, \cdots, r_l\rangle, \langle s_1 \cdots s_l\rangle]$, where $r_1 \cdots r_l$ is a sequence of l cloaks and $s_1 \cdots s_l$ is a sequence of l sets of LBS requests.

Example 16. The following are examples of bundles that use quad-tree based cloaks shown in the Figure 3.9(a) and the requests described in Example 15.

b_1 = (1, $\langle R_2, R_3\rangle$, $\langle\{v_1, v_3, v_5\}, \{v_2, v_4\}\rangle$)
b_2 = (2, $\langle R_3, R_3\rangle$, $\langle\{v_1, v_3, v_5\}, \{v_2, v_4, v_6\}\rangle$)
b_3 = (3, $\langle R_2, R_4\rangle$, $\langle\{v_3\}, \{v_4\}\rangle$)
b_4 = (4, $\langle R_3, R_4\rangle$, $\langle\{v_3\}, \{v_4\}\rangle$) □

At a high level, the function $cloak()$ represents a sequence of cloaks and the function $requests()$ represents a set of LBS requests sent along the cloak sequence, that is each LBS request in the set $requests(i)$ is sent from some location in the cloak $cloak(i)$. This allows us to use a bundle b to represent a set of user-history objects whose trajectories passes through all the cloaks of the bundle. Next we formally define this notion.

Definition 12. [**Masking**] Given a bundle b and a user-history u, we say the bundle b masks a user-history u if

a) (region containment): $\forall i$, $u.location(i) \in b.cloak(i)$; and
b) (information preservation): $\forall i$, $u.request(i) \subseteq b.requests(i)$

The requirement a) *region containment* describes the correctness condition for the coarser trajectory i.e. location of the user u at instance i must be contained within the region $cloak(i)$ of the bundle b. The requirement b) *information preservation* enforces the correctness condition for the set of request associated with the coarser trajectory. That is the set of LBS requests sent by the user u from location $u.location(i)$ must be a subset of the set of LBS requests sent from the region $b.cloak(i)$ of bundle b.

Example 17. The bundle b_1 in the Example 16 masks the users-history objects $Alice$, Bob, $Carrol$ and Sam of the Example 15. Similarly the bundle b_3 masks the user-history objects Bob and $Carrol$. □

The bundle object obfuscates the association of the LBS requests with the trajectory of the actual sender since for any two requests, $r_x \in requests(i)$ and $r_y \in requests(j)$, sent from two separate cloaks $cloak(i)$ and $cloak(j)$ in the sequence, where $i \neq j$, one cannot distinguish if they were sent by the same user. The high-level intuition here is that instead of publishing the set of user-history objects, the LBS publishes a set of bundle objects that masks the user-history objects. The bundle objects are so chosen that an attacker, who has access to the $location()$ function i.e. trajectories of all the users, cannot associate a request in the bundle with any one user thus preserving sender anonymity. We formalize this next.

4.3.2 Anonymizing Policy

Definition 13. [**Anonymizing Policy**] We define an *Anonymizing Policy* as a function that takes as an input a set U of users history objects and outputs a set B of bundles. Formally:

P : $\{$user history objects $\} \rightarrow \{$ bundle objects $\}$

A policy P, is a masking policy, if for each user-history object u, the bundle P(u) masks u. In this investigation we only consider masking policies, henceforth the term policy refers to a masking policy. Next we give example of a masking policy.

Example 18. The following anonymizing policy P_1 anonymizes the 5 user-history objects in Example 15 using the bundle objects shown in Example 16.

$$\begin{array}{ll} P_1(Alice) = b_1 & P_1(Bob) = b_3 \\ P_1(Carrol) = b_3 & P_1(Sam) = b_1 \\ P_1(Tom) = b_2 & \end{array}$$

□

As described in the section 3.1 using snapshot k-anonymization for anonymizing LBS request log does not provide sender k-anonymity because a LBS request sent by a user u in one instant is anonymized without considering the anonymization of his other LBS requests in other instants. The anonymizing policy takes into account all the requests sent by a user (via the user-history object) in the anonymization.

4.3.3 TP-aware Sender k-anonymity

Now we define the *TP*-aware sender k-anonymity guarantee. To do so we need to formalize the class of attackers who are aware of user trajectories and the anonymization algorithm.

Attacker Model. We target a strong information-theoretic definition of privacy therefore we model the attacker as a function taking certain input to launch the attack. There are no limiting assumption on the computation resources expended to launch the attack. The only assumptions are on what input the function has (intuitively, the information that the attacker sees). We classify the input into two groups as follows.

Design time: Even before the attacker observes the set of bundle objects, he may know

- the target level k of k-anonymity.
- the specific anonymizing policy P used to k-anonymize the users.

Run time: The attacker can obtain (via surveillance or subpoenas or financial agreement)

- the functions $userid()$ and $location()$ for all the $user - history$ objects (the trajectory of all the users).
- the published $bundle$ objects.

We refer to class of such attackers as *Trajectory-aware and Policy-aware* attackers. The attack function models the following attack: starting from the observation of a set S of bundle objects, the knowledge of trajectories of all the users and the anonymization policy, the attacker reverse engineers the bundle objects to obtain the possible user-history objects masked by S and compatible with the anonymizing policy i.e. user-history objects that are anonymized to bundle objects in S.

We are now ready to define *TP*-aware) sender k-anonymity. Intuitively, it captures the property that for each observed bundle object b there are at least k user-history objects that are anonymized to it. We consider it a breach of sender k-anonymity if for any bundle b the attacker succeeds in reducing the number of user-history objects that can possibly be anonymized to it, to less than k. We first define sender k-anonymity as a property of set of bundle objects w.r.t. a set of user-history objects and an anonymization policy. Since the bundles are obtained using a policy P, it is easy to extend the definition as a property of policy P.

Definition 14. [**TP-aware Sender k-anonymity**] Let P be an anonymization policy and U be a set of user-history objects. Let B be the set of bundles obtained using the policy P. We say B provides **TP-aware** sender k-anonymity if for each bundle $b \in B$ there are at least k distinct user-history objects in U that are anonymized to b under P.

We say that policy P provides *TP*-aware sender k-anonymity, if for every set of user-history objects U, the set of bundles $\{P(u)|u \in U\}$ provides *TP*-aware sender k-anonymity.

Using the above definition of *TP*-aware sender k-anonymity and the attack model described earlier, we show that the policy P_1 in Example 18 does not provides *TP*-aware sender 2-anonymity.

Example 19. The policy P_1 of Example 18 anonymizes the set of user-history objects $\{u_a, u_b, u_c, u_s, u_s\}$ shown in Example 15 to the set of bundles $\{b_1, b_2, b_3\}$ shown in Example 16. When the trajectory-aware policy-aware attacker observes b_1, he tries to reverse engineer the user-history objects that could have anonymized to it. He finds two user-history objects, u_a and u_s, corresponding to users $Alice$ and Sam. Therefore b_1 provides *TP*-aware sender2-anonymity. Similarly for b_3, there are 2 users u_b and u_c that could be anonymized to b_3. In contrast, when the attacker observes b_2, he can breach sender 2-anonymity since there is only one user-history object u_t that is anonymized to b_2 under P_1. Thus policy P_1 does not provide *TP*-aware sender 2-anonymity.

In the remainder of the paper we target an anonymization policy that preserves

TP-aware sender k-anonymity. We start with first showing an example policy that provides *TP*-aware sender 2-anonymity to the users of Example 15.

Example 20. For the five user-history objects in Example 15 we describe the following anonymizing policy P_2 that uses the bundle objects shown in Example 16.

$$\begin{array}{ll} P_1(Alice) = b_2 & P_1(Bob) = b_3 \\ P_1(Carrol) = b_3 & P_1(Sam) = b_2 \\ P_1(Tom) = b_2 & \end{array}$$

For both the observed bundle b_2 and b_3, there are at least 2 user-objects that can be anonymized to each one of them under policy P_2. When the trajectory-aware policy-aware attacker observes the published bundles, he tries to reverse engineer the user-history objects that could have anonymized to them. For both the bundles b_2 and b_3, he finds at least 2 users. Hence P_2 provides sender 2-anonymity against the trajectory-aware and policy-aware attackers. □

4.4 Optimum Anonymity

For the same set of user-history objects there may exist several anonymization policies that provide *TP*-aware sender k-anonymity, raising the obvious question of which one to use. In this section we address the problem of finding the k-anonymous policy of highest utility to the consumers of the published data. Prior work[DHVZ10, KGMP07, GG03, MCA06] on snapshot sender k-anonymity propose that one way to maximize utility is to minimize the area of the cloaks. For the log of LBS requests, an analogous measure would be to minimize the sum of the area of sequence of cloaks used in the bundles to anonymize the users.

Cost of a bundle. We introduce the cost of a bundle to quantitatively capture the fact that the utility is maximized, as the sum of the area of cloaks in the cloak sequence of the bundle is minimized. Given a bundle b = $[rid, \langle r_1 \cdots r_l \rangle, \langle s_1 \cdots s_l \rangle]$, we define the cost of b as the sum of area of cloaks in its cloak sequence.

$$Cost(b) = \sum_{i=1}^{i=l} area(r_i)$$

Given a collection U of user objects and an anonymizing policy P, we define the *cost* of a policy P for anonymizing U, denoted $Cost(P, U)$, as

$$Cost(P, U) = \sum_{u \in U} Cost(P(u))$$

4.4.1 Optimal policy

Using the cost model described above, one can quantitatively compare two anonymization policies and identify the one that has lower cost and therefore better utility. We next focus on the problem of finding the optimum (minimum cost) policy that provides *TP*-aware sender k-anonymity to a given set of user-history objects.

For policy-aware snapshot k-anonymity, it was shown in [DHVZ10] that the complexity of finding optimum policy depends upon the type of cloaks used for anonymization. In particular, finding optimum policy among all the policies that use circular cloaks is NP-hard [DHVZ10] whereas one can find a optimum policy that use quad-tree based cloaks in polynomial time (of the number of users) [DHVZ10]. We investigate whether similar results hold for the problem of finding optimum policy that provides *TP*-aware sender k-anonymity. We study this problem for two types of cloaks:

- Circular cloaks: when the cloak used in the cloak sequence of bundles are circular, and each cloak is centered at a point from a given set of points (e.g. public landmarks such as libraries, train stations or cell towers) with no restriction on the radius.
- Quad cloaks: when the cloaks used in the cloak sequence of bundles are picked among the quadrants of a quad-tree based partitioning of the map. The quad tree is a well-known structure for organizing spatial data, and it has been used in a number of anonymization solutions [GG03, MCA06, DHVZ10] for snapshot sender k-anonymity.

Anonymization using Circular cloaks. Let U be a set of user-history objects and SC be a set of points in the 2-dimensional space that contains the trajectories of the users. We define *circular cloak sequence* as a sequence of cloaks where each cloak is centered at some point from SC, with no restriction on the radius. Let $\mathcal{P}$ be the set of all those policies that use circular cloak sequence in the bundles for anonymizing user-history objects. The problem of *Optimum Offline* TP*-aware k-anonymization with Circular cloaks* is to find a policy in $\mathcal{P}$ that minimizes the cost of anonymizing U.

Theorem 3. Optimum Offline *TP*-aware k-anonymization with Circular cloaks is NP-hard.

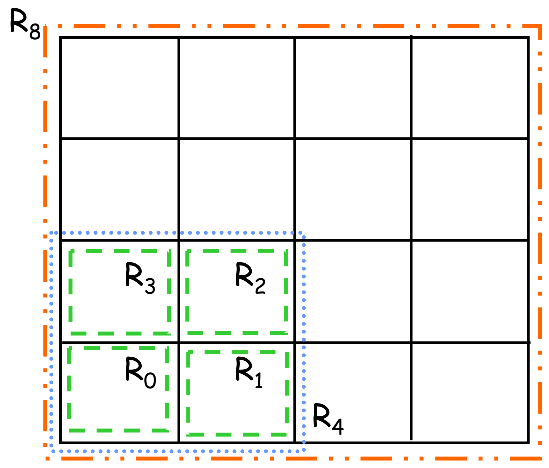


Figure 4.4: Quad-tree Partitioning

Anonymization using Quad-cloaks. Now we consider policies that uses cloaks picked from among the quadrants corresponding to a quad-tree-based partitioning of the geographic region. The root node of the quad-tree represents the entire region (square shaped) which is then partitioned into 4 equal non-overlapping square quadrants, each of whom represent a child node of the root. Each quadrant is then again divided into 4 equal sub-quadrants that correspond to grandchildren of the root. This four-way splitting goes on until the desired level of granularity for the minimum region is reached. Figure 4.4 shows a part of a quad-tree based partitioning: region R_8 represents a quadrant in the quad-tree that is divided into 4 equal sub-quadrants (e.g. R_4). The sub-quadrant R_4 is further divided into 4 equal sub-quadrants R_0, R_1, R_2, and R_3.

Given a quad-tree representation Q of a region, we refer to a sequence of cloaks, where each cloak is one of the quadrant of Q, as a *quad-cloak sequence*. For e.g. $\langle R_0, R_3 \rangle$ is a quad-cloak sequence of length 2 that uses the quadrants of quad-tree partitioning shown in Figure 4.4. A policy that anonymizes user-history object using bundles with quad-cloak sequence is referred to as a *quad-cloak policy*.

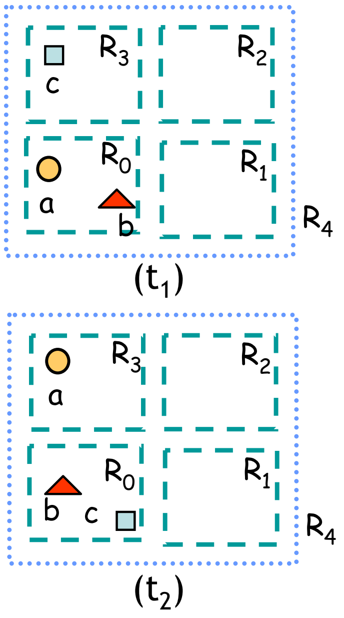


Figure 4.5: Trajectories of length 2

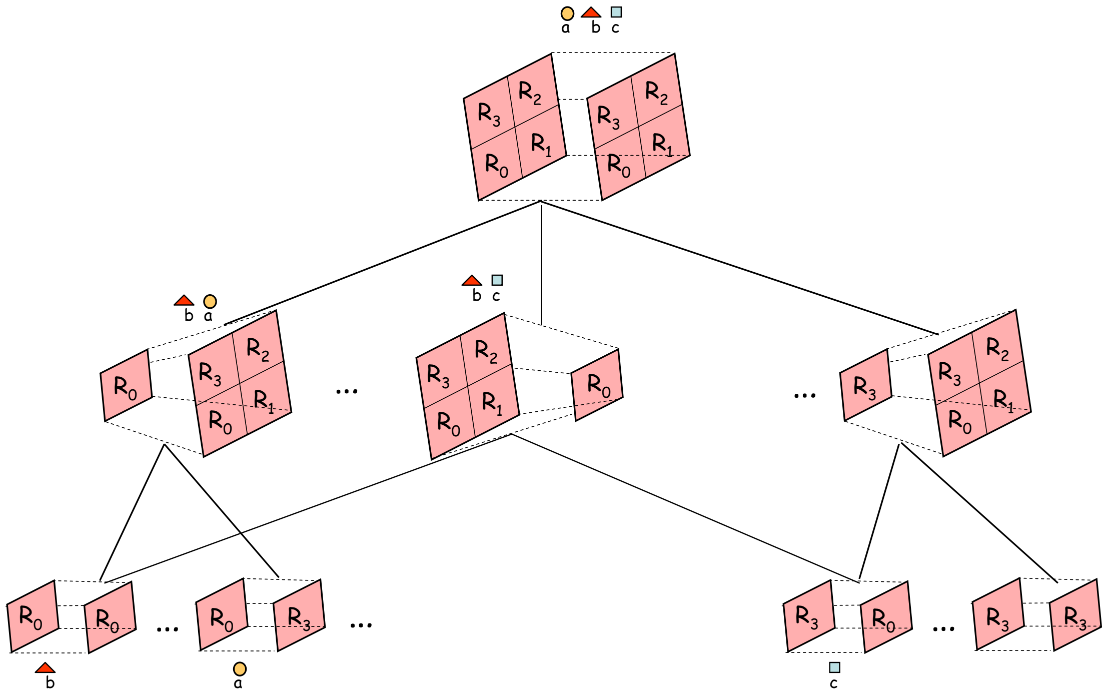


Figure 4.6: Trajectory Anonymization using Quad-tree

Optimum quad-cloak policy. Given a quad-tree Q and a set U of user-history objects, there exists several quad-tree policies that can be used to anonymize U. The number of quad-tree policies that can be used to anonymize U can be easily determined if one knows the number of quad-cloak sequences that masks the trajectory of a user-history object.

Let us assume the user-history objects in U are of length l and the quad-tree Q is of height h. For any location in the trajectory of a user-history object, there are h cloaks in Q that masks it (all the cloaks from leaf to root in Q). Therefore, for a trajectory of length l, there are h^l different quad-cloak sequences that mask the entire trajectory, representing the number of ways in which a user-history object can be anonymized using Q. If there are n user-history objects in U, then there are (h^{l^n}) or ($h^{(nl)}$) different ways of anonymizing them using quad-cloak sequences of Q. It also represents the number of different quad-tree policies that can possibly anonymize U, although not all of them provide *TP*-aware sender k-anonymity.

The problem of *optimum offline* TP-*aware sender k-anonymity with quad-cloaks* is to find a quad-cloak policy P that has the minimum cost of anonymizing U. Clearly a brute force search among all ($h^{(nl)}$) of the quad-tree policies to find the optimum that provides *TP*-aware sender k-anonymity cannot be done in polynomial time. As shown by our next result one cannot hope to do better i.e. find the optimum policy in polynomial time (unless P = NP).

Theorem 4. Optimum offline *TP*-aware sender k-anonymity with quad-cloaks is NP-Hard.

In hindsight the above result was expected given the role (curse) of dimensionality, since each location in the trajectory (acting as a variable) has (discrete) finite choices of cloaks. The combination gives rise to exponential number of choices of cloak sequence to anonymize a set of user-history objects.

4.4.2 Approximation Algorithm

The next best thing in lieu of an polynomial time optimum solution is to find a polynomial time approximation solution with bounded approximation factor. In this section, we show that the problem of optimum offline *TP*-aware sender k-anonymity with quad-cloaks is approximable and describe a polynomial time bounded approximation solution.

In our approach, we restrict the choices of cloak sequence that a policy can use, to a subset of all the possible choices of quad-cloak sequences. This amounts to identifying a subset S' of the set S of all the quad-cloak policies. The subset S' is chosen such that an optimum quad-cloak policy relative to S' can be found in polynomial time, and that this policy is a bounded approximation of optimum quad-cloak policy in S.

To simplify the description of a subset S' of all the quad-cloak policies and the algorithm that finds the optimum solution w.r.t. S', we utilize a structural relation that exists between quad-cloak sequences of a given length l. We refer to this relation as *1-step generalization*.

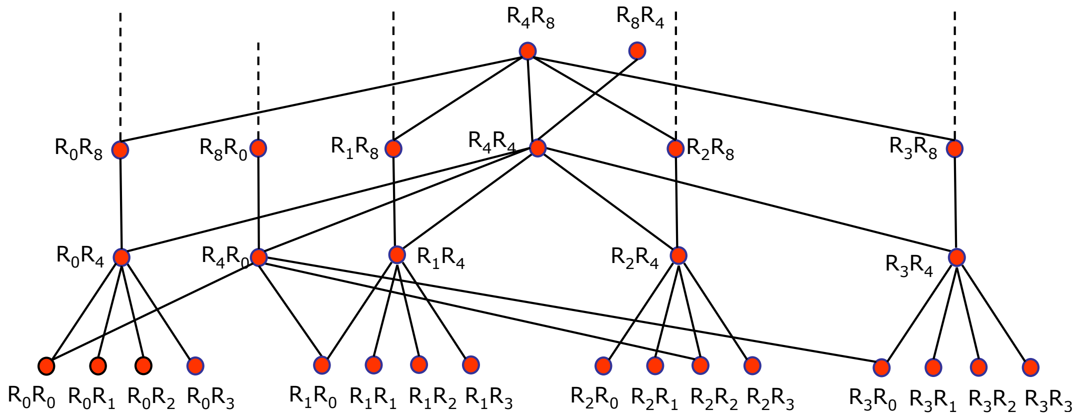


Figure 4.7: Subgraph of G-graph induced by 1-step generalization

1-step Generalization. Let Q be a quad-tree and s be a quad-cloak sequence of length l that uses quadrants of Q. Let s' be a quad-cloak sequence obtained by replacing one of the cloaks in s with its parent in Q. We refer to s' as *1-step generalization* of s. For each quad-cloak sequence of length l (with the exception of the quad-cloak sequences that contain the root quadrant of Q), there are l different 1-step generalizations. Also, each quad-cloak sequence of length l (with the exception of the cloak sequences that contain one ore more leaf quadrants of Q) is 1-step generalization of l quad-cloak sequences. The 1-step generalization relation induces a directed acyclic graph over all the quad-cloak sequences of a given length l obtained using a quad-tree Q. We refer to this graph as the *Generalization Graph*. Formally

Definition 15. [**Generalization graph**] Given a quad-tree partitioning **Q** of a region, a *Generalization graph* (G-graph) of length l is a directed acyclic graph G = ($\mathcal{V}, \mathcal{E}$) where

- $\mathcal{V}$ is the set of all possible quad-cloak sequences of length l that uses quadrants of **Q**, and
- $\mathcal{E}$ is the set of edges (s_i, s_j) where s_i, $s_j \in \mathcal{V}$ and s_j is the 1-step generalization of s_i.

Figure 4.7 shows a subgraph of the G-graph induced by 1-step generalization on the quad-cloak sequences of length 2 that uses quadrants of the quad-tree shown in Figure 4.4.

In a G-graph of length l it is easy to observe that a trajectory of length l masked by a quad-cloak sequence s is also masked by the 1-step generalization of s. We refer to this property as the *containment* property. The containment property states that any trajectory masked by a cloak sequence s is also masked by all the nodes on all the paths from s to root node of the G-graph. An an example, consider the trajectory of the user a shown in Figure 4.5. This trajectory is masked not only by the quad-cloak sequence $c = \langle R_0, R_3 \rangle$ shown in Figure 4.6, but also by those such as $\langle R_0, R_4 \rangle$ and $\langle R_4, R_4 \rangle$ that lies on all the paths from c to the root node of the G-graph.

Next, using a G-graph we restate our approach to find an polynomial time approximation solution to the problem of optimum offline TP-aware sender k-anonymity with quad-cloaks. Given a G-graph G of length l and a set U of trajectories, let P_G be the set of all the policies that uses quad-cloak sequences from G. The problem of optimum offline TP-aware sender k-anonymity with quad-cloaks is to find the optimum policy in P_G. Since this is NP-hard, we identify a subspace T of the G, and find the optimum policy in the set P_T of all the policies that use cloak sequences from T. The choice of T is such that the optimum policy can be found in polynomial time and it is a bounded approximation of the optimum policy in P_G. In the next section we describe the characteristics of such a subspace.

Generalization Tree. Given a G-graph **G**, we define a **Generation tree** (G-tree) as a tree **T** in which every node has bounded degree and that preserves the ancestor-descendant relationship between nodes as observed in **G**. Formally, a generalization tree **T** of a G-graph G is defined as:

a) the nodes of T are a subset of the nodes in G.
b) If y is the parent of x in T, then y must be a ancestor of x in G.
c) each node in T has a finite bounded degree (i.e. each non-leaf node has finite number of children, known apriori).

The condition a) and b) ensure that a G-tree identifies a subspace of a G-graph. In other words the set $\mathcal{P}_T$ of all the policies that use the cloak sequences in T is a subset of all the quad-cloak policies $\mathcal{P}_G$. In addition, property b) above also preserves the containment property of G-graph in the corresponding G-tree. As a result any trajectory masked a node in T is also masked by its parent in T. As described next, this property along with condition c) is important in finding a polynomial time approximation solution.

Note that using the above definition, one can obtain multiple G-trees corresponding to a G-graph. The choice of a G-tree dictates the bounded approximation factor and the complexity of the algorithm that achieves the bound. We address the issue of identifying a G-tree with bounded approximation factor in Section 4.4.2. We first describe a generic algorithm that takes as input a G-tree T and finds the optimum policy w.r.t. to $\mathcal{P}_T$ for anonymizing a set of trajectories U in polynomial time (polynomial in number of trajectories in U and number of quad-cloak sequences in T).

Algorithm

To obtain a polynomial time algorithm that finds an optimum policy that uses quad-cloak sequences from a *G-tree* T, we utilize two unique properties of such policies. Using the first property we define equivalence classes of policies such that all the policies in an equivalence class have the same cost and anonymize same number of trajectories to each quad-cloak sequence in T. This allows us to reduce the problem of finding the optimum policy to finding the equivalence class corresponding to the optimum policy. Even though there are fewer equivalence classes than the policies, the total number of choices is still exponential (in number of cloak sequences in T). The second property allows us to use a divide and conquer strategy to prune and search the exponential search space of equivalence classes and find the one corresponding to the optimum policy.

Property 1: Cost of a policy is determined by the number of trajectories anonymized to each node in T . For a policy in $\mathcal{P}_T$, the property of being *TP*-aware sender k-anonymous and the cost of the anonymization depends upon *how many* trajectories are anonymized by each node in T, being indifferent to *which* particular trajectories are anonymized. Our next example demonstrate this observation.

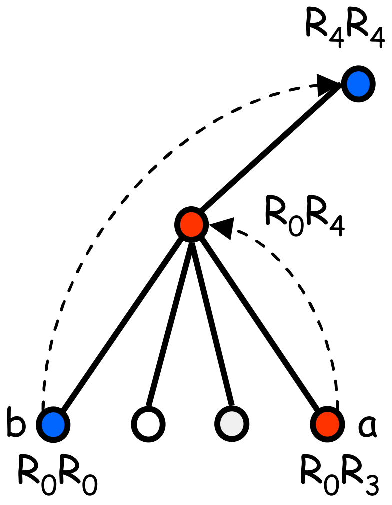


Figure 4.8: Policy P_1

Example 21. Consider trajectories a and b shown in Figure 4.5. Figure 9 and Figure 10 show the anonymization of these trajectories under two policies P_1 and P_2. Policy P_1 anonymizes a to cloak sequence $\langle R_0R_4\rangle$ and b to $\langle R_4R_4\rangle$, where as policy P_2 anonymizes a to cloak sequence $\langle R_4R_4\rangle$ and b to $\langle R_0R_4\rangle$. Except for this difference, all the other trajectories are anonymized identically in P_1 and P_2. Since

$Cost(P_1(a)) = Cost(P_2(b))$ and
$Cost(P_1(b)) = Cost(P_2(a))$

and therefore

$$Cost(P_1(a)) + Cost(P_1(b)) = Cost(P_2(b)) + Cost(P_2(a))$$

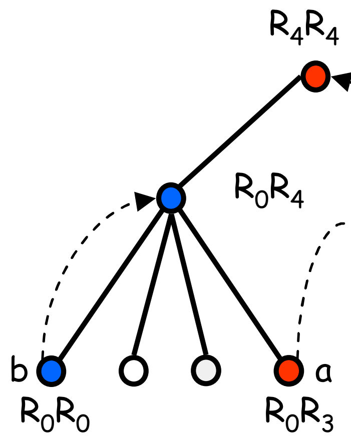


Figure 4.9: Policy P_2

as a result, the cost of two policies P_1 and P_2 is identical. $\square$

We formalize this observation as an equivalence relation among policies in $\mathcal{P}_T$ that use quad-cloak sequences in G-tree T. Two policies in $\mathcal{P}_T$ are equivalent for a given set of trajectories if every node in G-tree T anonymizes the same number of trajectories under both policies.

Lemma 6. If policies P_1, P_2 are equivalent for a G-tree T, then

(a) P_1 and P_2 have the same cost; and

(b) P_1 provides *TP*-aware sender k-anonymity on T if and only if so does P_2.

Using this equivalence between a set of policies we reduce the problem of finding optimum policy in $\mathcal{P}_T$ that provides TP-aware sender k-anonymity, to the problem of finding the equivalence class of that optimum policy. Next we formally define this equivalence class using a *Configuration* function.

Configuration. The function *Configuration* is defined to keep track of the number of trajectories anonymized by each node m in a G-tree T. For technical convenience, this is done by equivalently tracking for each node m the number of trajectories that are masked by m yet are not anonymized using m or any of its descendants. We refer to these trajectories as *passed up*. It is easy to translate between the two equivalence class representations.

Definition 16. [**Configuration**] Let U be a set of trajectories and T be a G-tree rooted at r. Let $d(m)$ denote the total number of user trajectories that are masked by the cloak sequence represented by node m. A *Configuration* C is a function from nodes of T to natural numbers, such that

(i) for every leaf node m, $C(m) \leq d(m)$; and
(ii) for every internal node q, $C(m) \leq \sum_{i=1}^{f} C(m_i)$, where m has f children $m_1, \ldots$, m_f.

We say that C is $complete$ if $C(r)$ = 0. □

Condition *(i)* in the above Definition 16 restricts a configuration to represent only masking policies and *(ii)* represents the fact that a trajectory can be anonymized to only one cloak sequence. Note that under our cost function, all the policies in the equivalence class represented by a configuration C have the same cost. We call this the cost $Cost_c$ of the configuration C. We can compute this cost directly using the configuration and without enumerating any policy.

Definition 17. [**Configuration cost**] Let U be a set of trajectories and C be a configuration of the G-tree T. We define the *cost of C for U*, denoted $Cost_c(C, U)$, as

$$Cost_c(C, U) := \sum_{m \in nodes(T)} f(m, C) \times Cost(m)$$

where $f(m, C)$ is given by

$$f(m, C) = \begin{cases} (d(m) - C(m)), & \text{If m is leaf} \\ ((\sum_{i=1}^{l} C(m_i)) - C(m)), & \text{If m is internal} \end{cases}$$

where $m_1 \ldots m_l$ are the children of m and Cost(m) is the sum of area of the cloaks in cloak sequence corresponding to node m. □

We can show that the configuration cost is precisely the cost of the represented policies:

Lemma 7. Given a set U of n trajectories of length l, a tree T of quad-cloak sequences of length l, a policy P that use cloak sequences from T and a configuration C representing P's equivalence class, we have

$$Cost_c(C, U) = Cost(P, U)$$

.

Thus finding the optimum quad-tree policy that uses cloak sequences from T to anonymize a set U of trajectories is equivalent to finding optimum configuration C of the tree *T* w.r.t. U. Our algorithm does exactly that, i.e. we first find the minimum cost configuration and then materialize a policy corresponding to the minimum cost configuration, in polynomial time.

In addition to showing that finding the minimum cost configuration is equivalent to finding minimum cost policy, we need to ensure that this configuration corresponds to *TP*-aware sender k-anonymous policies.

Checking Sender Anonymity from Configurations. We turn to checking if the policies in the equivalence class represented by a given configuration are *TP*-aware sender k-anonymous, without materializing them. By Lemma 6(b), either all represented policies qualify, or none does. It turns out that it suffices to check directly that the configuration satisfies a property we call *k-summing*.

Definition 18. [**k-summing**] Let U be a set of trajectories and C a configuration of the tree T rooted at r. C is a k-summing configuration if

- for a leaf node m
  - (i) if $d(m) < k$, then $C(m) = d(m)$.
  - (ii) if $d(m) \geq k$, then either $C(m) = d(m)$ or $C(m) \leq (d(m) - k)$.
- for an internal node m let $\Delta = \sum_{i=1}^{l} C(m_i)$, where $m_1 \ldots m_l$ are the children of m in T
  - (iii) if $\Delta < k$, then $C(m) = \Delta$.
  - (iv) if $\Delta \geq k$, then either $C(m) = \Delta$ or $C(m) \leq (\Delta - k)$.

Intuitively, in Definition 18, clause (i) states that if quad-cloak sequence corresponding to node m masks less than k trajectories, none of them can be anonymized by m lest k-anonymity be compromised. The responsibility of anonymizing all $d(m)$

of them is passed up to m's ancestors ($C(m) = d(m)$). By clause (ii), if there are at least k trajectories, then either all of them are passed up, or at most $d(m) - k$ (since at least k must be anonymized to the same cloak sequence to preserve k-anonymity). For an internal node m, Δ represents the number of trajectories whose anonymization responsibility is passed up from m's children to m. If there are too few of them (less than k) then they cannot be anonymized using cloak sequence of m, who in turn passes the responsibility to its ancestors (in clause (iii)). Otherwise, m has the choice of either anonymizing none of them ($C(m) = \Delta$ in clause (iv)), or anonymizing at least k and passing up at most $\Delta - k$.

Next we show that the k-summing configuration is equivalent to all its represented policies being *TP*-aware sender k-anonymous.

Lemma 8. Let T be a G-tree of quad-cloak sequences and U be a set of trajectories. Let C be a configuration of T for anonymizing U and P be a policy in the equivalence class C represents. P provides *TP*-aware k-anonymity to U if and only if C is a k-summing configuration.

Lemmas 7 and 8 justify an algorithm that explores the space of k-summing configurations, in search for a complete minimum-cost configuration. But for a set of n trajectories and a G-tree T with m nodes there are $O(n^m)$ possible configurations. Next we describe the second property of the policies in $\mathcal{P}_t$ that enables a divide and conquer approach to find the optimum k-summing configuration.

Property 2: Optimum cost of anonymizing a subset of trajectories using a node in T can be computed locally. Let C be a k-summing configuration C of a G-tree T of quad-cloak sequences. For a node m in T, $C(m)$ represents the number of unanonymized trajectories passed up by m. These passed up trajectories are anonymized at one of the ancestors of m and hence affects the anonymization cost of other trajectories masked by the ancestors of m. But they do not affect how the $d(m) - C(m)$ trajectories are anonymized using m and its descendants. Thus for a given value of $C(m)$, one can optimize the anonymization of $d(m) - C(m)$ trajectories using m and its descendant independent of the rest of the trajectories. Before we describe how we compute this local optimum for each m, we need to point out that at this stage we don't know the value of $C(m)$ in the optimum configuration. For this reason we compute the optimum costs of passing up $0, 1 \ldots d(m)$ trajectories i.e. all possible values of $C(m)$. For each such pair (m, u), the minimum cost is computed among all possible configurations of T_u where $C(m) = u$ (as there are many possible configurations with $C(m) = u$).

Computing all local optimum costs. To compute the (local) optimum value of passing up u trajectories at node m, the algorithm considers all possible counts $\langle 0, 1 \ldots d(m_1) \rangle$, $\langle 0, 1 \ldots d(m_2) \rangle$, ..., $\langle 0, 1, \ldots, d(m_f) \rangle$ of trajectories passed by its children $m_1, \ldots, m_f$ respectively. Then it recursively computes the corresponding minimum cost for each (m_i, u_i) pair. Redundant cost re-computation for m, u pairs is avoided by storing the result in the corresponding cell of a bi-dimensional matrix M indexed by the nodes of T_u and values of u. To enable the easy retrieval of min-cost configuration from M, the entries for node m carry, besides the minimum cost, some bookkeeping information relating to the configurations of the children of m.

This yields the following dynamic programming algorithm *Traj-anon* that, given a set U of trajectories of length l and a tree T with cloak sequences of length l, fills in a configuration matrix M of dimension $|T| \times |U|$, where $|T|$ represents the number of nodes in T and $|U|$ the number of trajectories in U. Each entry $M[m][u]$ in the matrix is a tuple of the form $\langle x, u_1, u_2, \ldots, u_f \rangle$, pertaining to a configuration C such that $C(m) = u$, and where x is the minimum cost of passing up u trajectories, with the child nodes $m_1, m_2, \ldots, m_f$ of m passing up $u_1, u_2, \ldots, u_f$ trajectories respectively. The algorithm traverses the tree T bottom-up starting from the leaf nodes, and for each node and $1 \leq u \leq d(m)$ fills in the entry $M[m][u]$ using the rows from child nodes $m_1, m_2, \ldots m_f$.

Selecting the optimum configuration. The optimum configuration is obtained when the optimum cost of $C(r) = 0$ is computed, where r is the root node of T. After that it is easy to retrieve the complete configuration from M in polynomial time by a top-down traversal of T. The minimum cost entry $M[r][0]$ for root r lists for its each child m_i the value $C(m_i) = u_i$ leading to the minimum cost. Now inspect for each m_i the u_i entry in M, picking again the minimum cost entry for passing up u_i trajectories at m_i and continue recursively untill all leaf nodes are reached.

Function $F(m)$ in line 16 limits the possibilities of the number of trajectories whose anonymization responsibility can be passed up by m. Notice that it rules out the values $d(m) - k + 1$ through $d(m) - 1$ since these imply anonymizing less than k trajectories at m, which would immediately compromise k-anonymity. Quantity x is the minimum cost among all configurations C with $C(m) = u$ and which satisfies k-summation property. This is computed from the costs of the configurations at the f children, and the number of trajectories anonymized by m i.e. $((\sum_{l=1}^{f} u_l) - u)$. Recall that the cost is the first component of the tuple stored in the matrix entry, whence the need for the projection operation M^1.

Notice how the algorithm mirrors the definition (Definition 18) to ensure that only k-summing configurations are considered. By Lemma 8, these configurations represent only *TP*-aware sender k-anonymous policies. For instance, line 8 corresponds to case (i) in Definition 18, which prescribes that no trajectories are to be anonymized by m (all $d(m)$ trajectories inside the cloak sequence of m are passed up, $C(m) = d(m)$). Thus by Definition 17, the resulting cost is 0, which is what line 8 fills into the first component of $M[m][d(m)]$. Similarly, line 10 gives the cost corresponding to the case in the first disjunct of line (ii) of Definition 18; line 12 corresponds to the second disjunct. It's easy to see that:

Lemma 9. *Algorithm* **Traj-anon** *computes in each* $M[m][u] = \langle x, u_1, u_2, \ldots, u_f\rangle$ *the configuration with minimum cost* x *among all k-summing configurations* C *where* $C(m) = u$ *and where* $C(m_i) = u_i$*, with* $m_1, \ldots, m_f$ *the children of* m*.*

By the above discussion, the information in M suffices to retrieve in PTIME a minimum-cost configuration.

Complexity analysis. The running time of Algorithm *Traj-anon* is dominated by steps 16-18. For internal node m, it ranges each of $u, u_1, u_2, \ldots, u_f$ over at most $|U|$ values (since $F(m) \leq d(m) \leq |U|$ for every m), resulting in $O(|U|^f)$ iterations where the degree f represents the maximum number of children of a node $m \in T$. Summing up over all nodes m of the tree subspace T, we obtain the complexity of *Traj-anon* in

Algorithm 2 Traj-anon

```
for $1 \leq m \leq |T|$ do
  for $1 \leq u \leq |U|$ do
    M[m][u] := $\langle \infty, 0, 0, 0, 0 \rangle$ {initialize}
  end for
end for
for all node $m \in T$ do
  if (m is a leaf node) and (d(m) < k) then
    M[m][d(m)] := $\langle 0, 0, 0, 0, 0 \rangle$
  else if (m is a leaf node) and (d(m) $\geq$ k) then
    M[m][d(m)] := $\langle 0, 0, 0, 0, 0 \rangle$
    for $0 \leq u \leq d(m) - k$ do
      M[m][u] := $\langle area(m) \times (d(m) - u), 0, 0, 0, 0 \rangle$
    end for
  else {m is a non-leaf node}
    let $m_1, m_2, \ldots, m_f$ are children of m
    for all u in F(m) do
      pick $u_1 \in F(m_1), u_2 \in F(m_2)$, ..., $u_f \in F(m_f)$
      that minimize the quantity
      $x := \sum_{l=1}^{f} M^1[m_l][u_l]$ + $(sarea(q) \times ((\sum_{l=1}^{f} u_l) - u))$
      where
      $F(m)$ denotes the set $[0..(d(m) - k)] \cup \{d(m)\}$,
      and $M^1[i][j]$ returns the first component of the
      tuple at $M[i][j]$
      M[m][u] := $\langle x, u_1, u_2, \ldots, u_f \rangle$
    end for
  end if
end for
return M
```

$(O|T||U|^f)$. The algorithm is polynomial time where the degree f of the polynomial depends upon the chosen tree T of G-graph.

Algorithm 3 $Config2Policy$

```
for all node $m$ of T-uniform do
  $m.tempgroup \leftarrow []$
end for
for all node $m$ of T-uniform from bottom up do
  if m is a leaf node then
    $m.tempgroup \leftarrow$ trajectories within $m$
  end if
  $x \leftarrow$ the number of trajectories to be passed up from $m$
  $s \leftarrow$ the subsequence of last $x$ trajectories in $m.tempgroup$
  Remove the last $x$ trajectories in $m.tempgroup$
  Append $s$ to the end of $m.parent.tempgroup$
  Trajectories in $m.tempgroup$ are anonymized by cloak sequence $m$
end for
```

Policy from Configuration. We do not enumerate all the policies of the equivalence class corresponding to the optimum configuration. Note that a C configuration is exponentially more succinct than an explicit listing of the policies it represents; if we focus on any node m alone, there are exponentially many ways to pick $C(m)$ trajectories among those occurring in m. Yet, we can obtain one of the policies C represents in linear time by non-deterministically selecting the $C(m)$ trajectories for each node m as follows. After an optimal configuration is computed, we can create a concrete policy by creating groups of trajectories that are anonymized to nodes of the tree T. This is guided by the number of trajectories that should be passed up from the optimal configuration. Algorithm 3 gives the pseudo-code of this procedure. It performs a bottom-up traversal of T and for each node, it creates a temporary list *tempgroup* that holds all the trajectories passed up from the children. For a leaf node, this array contains the trajectories that are within the cloak sequence. The trajectories to be passed up to the parent are selected from the end of the *tempgroup*. The remaining trajectories are anonymized by the node.

l-approximation

Our approach for finding an approximation solution to the problem of optimum *TP*-ware sender k-anonymity using quad-tree policy consists of *a)* identifying a subset S' of all the possible quad-cloak sequences and *b)* finding the optimum policy among those policies that only uses the cloak sequences from S'.

In the previous subsection we described an algorithm *Traj-anon* that can find the optimum policy w.r.t. a tree subspace of the G-graph of quad-cloak sequences. In this section we describe a subset of quad-cloak sequences that defines a tree subspace T, and show that the optimum policy w.r.t. T is a bounded approximation of the over all optimum policy. Our first tree subspace T_u is obtained by limiting the choice of cloak sequences to *uniform cloak sequences*.

Uniform Cloak-Sequence Tree. Let D be a G-graph of quad-cloak sequences of length l that use quadrants of a quad-tree Q. Consider a quad-cloak sequence in D in which all the cloaks are of the same size. We refer to such a cloak sequence as *uniform quad-cloak sequence*. The cloak sequences $\langle R_0, R_0\rangle$, $\langle R_0, R_3\rangle$ and $\langle R_4, R_4\rangle$, shown in Figure 4.6, are examples of uniform quad-cloak sequence. Let s be a uniform quad-cloak sequence in D. Let s_p be the cloak sequence obtained by replacing each cloak in s with its parent in Q. We refer to s_p as the *total 1-step generalization* of s. Thus total 1-step generalization is a well defined function over the set of quad-cloak sequences. We show that the subset S' consisting of uniform quad-cloak sequence from D and the total 1-step generalization function defines a tree T_u as follows.

- each uniform quad-cloak sequence in D is a node in T_u.
- If s_p is the total 1-step generalization of $s \in T_u$, then $s_p \in T_u$ and we set s_p as the parent of s.

The tree T_u is a subspace of G-graph D since it has the following properties.

- the nodes in T_u are a subset of the nodes in D, since the uniform quad-cloak sequences in T_u are selected from D.

- the total 1-step generalization s_p of s is obtained by replacing all the cloakes in s with their parents in Q, therefore s_p is an ancestor of s in D.
- each quad-cloak sequence in T_u has a unique total-step generalization. In other words, every node in T_u has only one parent, which makes T_u a tree.
- Since every quadrant in Q have 4 child nodes, there exists 4^l uniform quad-cloak sequences of length l that have the same total 1-step generalization. Thus each node in T_u has finite bounded degree (4^l).

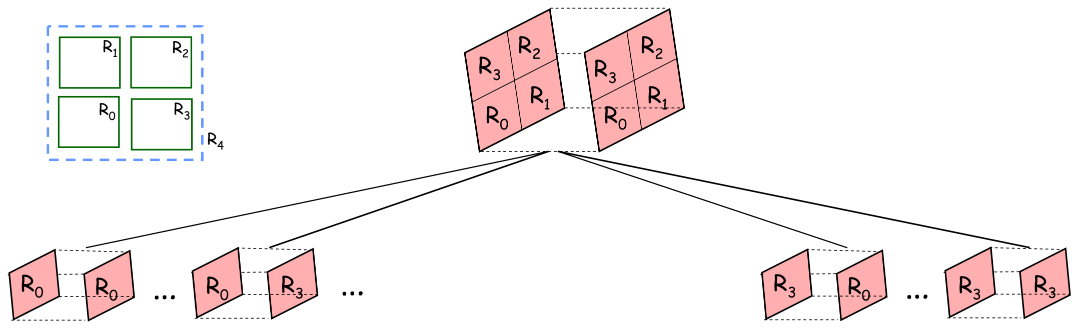


Figure 4.10: Uniform Cloak-sequence Tree

Figure 4.10 shows an example of such a tree where each node is a uniform cloak sequence that uses quadrants from the quad-tree shown in the left of the tree. Note that its nodes are subset of G-graph shown in Figure 4.6, and the parent node $\langle R_4, R_4 \rangle$ is obtained using total 1-step generalization of the child nodes (e.g. $\langle R_0, R_0 \rangle$). Also the parent $\langle R_4, R_4 \rangle$ and its child nodes have ancestor-descendant relationship in G-graph. Since the length of the cloak sequences is $l = 2$, there exists 4^2 (bounded) child nodes for each parent node in this tree.

We refer to this tree as the *Uniform Cloak-Sequence Tree* (T-uniform) since it includes only and all the uniform quad-cloak sequences of the G-graph. The root of *T-uniform* is the sequence of quad-cloak corresponding to root of Q. The leaf nodes are the uniform cloak sequences where each cloak is a quad-cloak corresponding to a leaf node of Q.The intermediate nodes are uniform quad-cloak sequences where each cloak is a quad-cloak corresponding to a non-leaf node of Q. The height of *T-uniform* is same as Q i.e. h since for each leaf uniform cloak sequence $h - 1$ successive total 1-step generalizations leads to the root uniform cloak sequence.

Uniform policies. A policy that only uses uniform quad-cloak sequence in the bundles is referred as *uniform* quad-cloak policy. Note that in a uniform quad-cloak policy, the cloaks need not be of same size across different cloak sequences.

Let U be a set of user trajectories and Q be a quad-tree. Let $\mathcal{P}_u$ be the set of all the uniform quad-cloak policies that anonymizes U using the quadrants from Q. As shown in Theorem 5 the best policy (lowest cost) among $\mathcal{P}_u$ that provides *TP*-aware sender k-anonymity is an approximation of the optimum quad-cloak policy that provides *TP*-aware sender k-anonymity to U.

Theorem 5. Given a set U of trajectories of length l, a quad-tree Q and degree of anonymity k, the cost of the optimum uniform quad-cloak policy that provides *TP*-aware sender k-anonymity is at most l-times that of optimum quad-cloak policy that provides *TP*-aware sender k-anonymity.

Therefore to anonymize a set U of trajectories of length l, we use the Traj-anon algorithm to obtain the optimum configuration w.r.t. T-uniform T_u of length l and U. The obtained configuration represent the equivalence class of policies that have the optimum cost among all the uniform policies, for anonymizing U. Due to Lemma 9 and Theorem 5, we can claim the following.

Theorem 6. Algorithm **Traj-anon** computes the l-approximation solution to the problem of optimum offline TP-aware sender k-anonymity.

Complexity analysis. As described earlier the complexity of Algorithm *Traj-anon* depends upon the maximum number of children of a node in chosen tree subspace. For a uniform tree T_u of length l, each non-leaf node has 4^l child nodes. Hence the complexity of *Traj-anon* is $(O|T_u||U|^{4^l})$. Even though the algorithm is polynomial time, the degree 4^l is impractically high as, in practice we expect long and large number of trajectories. In the next section we describe our optimization techniques to reduce the complexity of the *Traj-anon* on T_u without increasing the bounded cost.

4.5 Optimizations

In this section we describe optimizations to reduce the complexity of Traj-anon algorithm for T-uniform without affecting the upper bound on the cost i.e. the optimized algorithm is also an l-approximation of the optimum.

As mentioned in the section 4.4.2 the complexity of Traj-anon is determined by the maximum number of child nodes in a G-tree. In the case of a T-uniform T_u with cloak sequences of length l, each non-leaf node has (4^l) child nodes and hence the complexity of Traj-anon for T_u is $(O|T_u||U|^{4^l})$. Our first optimization reduces the number of maximum number of child nodes in T_u without eliminating any nodes from T_u thus reducing the complexity of Traj-anon without affecting the approximation factor.

4.5.1 USeq-Quad tree

Given a T-uniform tree T_u of uniform cloak sequences of length l, we introduce intermediate nodes (cloak sequences of length l) between a non-leaf node $m \in T_u$ and it's child nodes such that the resulting structure has the following properties:

- it is a G-tree .
- has all the nodes of T_u (and some additional nodes).
- y is an ancestor of x in this G-tree, if y is parent of x in T_u.
- each non-leaf node in this G-tree has exactly 4 child nodes.

We refer to this G-tree as *USeq-Quad tree*. The nodes that are inserted between a node $m \in T$ and its children are not uniform cloak sequences and are obtained by *ordered* 1-step generalization, that we describe next.

Ordered 1-step generalization. As described earlier, there are l 1-step generalizations of a cloak sequence of length l, one corresponding to each cloak in the cloak sequence. *Ordered 1-step generalization* refers to the process of obtaining l sequence of cloaks by l "successive" 1-step generalizations, such that the i_{th} 1-step generalization is obtained by replacing the i_{th} cloaks in the cloak sequences obtained by $(i-1)_{th}$ 1-step generalization.

Given a T-uniform T_u of length l, we obtain the USeq-Quad tree by inserting intermediate nodes, between node m and its 4^l child nodes, that are obtained by ordered 1-step generalizations of the child nodes. For each child node, we obtain l cloak sequences using ordered 1-step generalization and during the computation of ordered 1-step generalization, the cloak sequence obtained by i_{th} 1-step generalization is made a parent of the $(i-1)_{th}$ 1-step generalization. Each each child node, we obtain the node m in the l^{th} 1-step generalization.

We use only one node to represent a cloak sequence even if is obtained via total 1-step generalizations or two or more nodes. For all the child nodes, consider the 1_{st} 1-step generalization in the ordered 1-step generalization. Since each quadrant in Q has 4 child nodes, there are 4 child nodes that have identical 1_{st} 1-step generalization. Thus each of these intermediate nodes has 4 child nodes. Similarly each cloak sequence obtained in the i_{th} 1-step generalization is common for 4 intermediate nodes that were obtained $(i-1)_{th}$ 1-step generalization. In the l^{th} 1-step generalization we obtained the node m from the 4 intermediate nodes obtained by $(l-1)^{th}$ 1-step generalization.

Thus even after inserting the intermediate nodes, the resulting structure is a USeq-Quad tree, containing all the nodes of T_u, where each non-leaf node has exactly 4 child nodes. Moreover since the parent cloak sequences a 1-step generalization of its child, the parent nodes in G-tree completely masks their child nodes.

Next we adapt the Traj-anon algorithm to find the optimum configuration C for for a given USeq-Quad tree T_{usq} and a set U of users. C represents the equivalence class of policies that has the optimum cost among policies that use cloak sequences from T_{usq} to anonymize the set U of user-history objects. Moreover the cost of these policies is never worse then the cost of optimum policy that uses only uniform cloak sequences i.e. nodes in T_u.

Since T_{usq} is a quad-tree the complexity of the above algorithm dominated by the steps 16-18 is O($|T_{usq}||U|^5$) where $|T_{usq}|$ represents the number of nodes in $T_{usq}$and $|U|$ represents the number of user-history objects. Note that ordered 1-step generalization inserts $O(4^l)$ nodes between a non-leaf node and its children in T_u, therefore the resulting USeq-Quad tree T_{usq} has $O(m * 4^l)$ more nodes then T_u but this number is independent of the number of trajectories hence a constant factor. The reduced complexity results in reduced running time for finding the optimum configuration as observed in our experiments.

Algorithm 4 Modified Traj-anon

```
1: for $1 \leq m \leq |$ QT-super$|$ do
2:     for $1 \leq u \leq |U|$ do
3:         M[m][u] := $\langle \infty, 0, 0, 0, 0 \rangle$ {initialize}
4:     end for
5: end for
6: for all node $m \in$ QT-super do
7:     if (m is a leaf node) and (d(m) $<$ k) then
8:         M[m][d(m)] := $\langle 0, 0, 0, 0, 0 \rangle$
9:     else if (m is a leaf node) and (d(m) $\geq$ k) then
10:        M[m][d(m)] := $\langle 0, 0, 0, 0, 0 \rangle$
11:        for $0 \leq u \leq d(m) - k$ do
12:            M[m][u] := $\langle area(m) \times (d(m) - u), 0, 0, 0, 0 \rangle$
13:        end for
14:    else {m is a non-leaf node}
15:        let $m_1, m_2, \cdots, m_4$ are children of m
16:        for all u in F(m) do
17:            pick $u_1 \in F(m_1), u_2 \in F(m_2), \cdots, u_4 \in F(m_4)$
               that minimize the quantity
18:            $x := \sum_{l=1}^{4} M^1[m_l][u_l]$ + $(sarea(m) \times ((\sum_{l=1}^{4} u_l) - u))$
19:            where
               $F(m)$ denotes the set $[0..(d(m) - k)] \cup \{d(m)\}$,
               and $M^1[i][j]$ returns the first component of the
               tuple at $M[i][j]$
20:            M[m][u] := $\langle x, u_1, u_2, \cdots, u_4 \rangle$
21:        end for
22:    end if
23: end for
24: return M
```

Even though Traj-anon for a USeq-Quad tree of length l has reduced complexity in comparison to Traj-anon for a T-uniform of length l, we do trade cost to achieve better complexity. This is shown by the following result.

Lemma 10. Given a USeq-Quad tree T_{usq} of length l, T-uniform T_u of length l and a set U of user-history objects of length l and the level of anonymity k, the cost of the optimum k-summing configuration for T_{usq} is never more than the cost of the optimum k-summation configuration for T_u.

As a result the upper bound on the approximation ration i.e l still holds for a USeq-Quad tree. This follows directly from Lemma 10.

Theorem 7. Algorithm Modified Traj-anon computes the l-approximation solution to the problem of optimum offline TP-aware sender k-anonymity.

Moreover the average case cost of optimum policy that uses T_{usq} is lower than the policy that uses T_u since when using T_{usq} a policy has more options for choosing a sequence of cloaks to anonymize a user object.

While the above optimization leads to polynomial time algorithm with constant exponent, the degree 5 is still high given the typical number of LBS users in a metro city (in the range of 1 million for city like San Francisco). Next we describe a pair of optimizations to further reduce the complexity of Traj-anon and run-time optimizations to achieve practical running time, while guaranteeing to preserve the approximation bound. These optimizations are inspired by the optimizations for $Bulk_{dp}$ described in Section 3.5.

4.5.2 From Quad to Binary Tree

In the USeq-Quad-tree, if anonymizing a trajectory to a node does not provide the desired k-anonymity, the next possible option is the parent node. Since the parent node is 1-step generalization of the child, the cost of the new cloak in the parent cloak sequence is 4 times that of the replaced cloak in the child cloak sequence. As shown first in [MCA06] and also in Section 3.5, the granularity of this cost increase can be reduced by converting a quad-tree into a binary tree by using *semi-quadrants* as cloaks (where a semi-quadrant is obtained by splitting a quadrant into two rectangles, either vertically or horizontally). Use semi-quadrants in the uniform cloak sequences to anonymize the trajectories leads to following:

- The Traj-anon algorithm finds the optimum policy among all the policies that use uniform semi-quadrant cloak sequences.
- The policy obtained using Traj-anon is l-approximation of the optimum policy that use semi-quadrant cloak sequences.
- Using total 1-step generalization we can obtain USeq-Btree (analogous to USeq-Quad tree) that contains all the uniform semi-quadrant cloak sequences, and in which each node has exactly 2 child nodes.

As a result of this optimization, the complexity of the optimized Traj-anon for a USeq-Btree T_{usqb} and a set of trajectories U, is O($|T_{usqb}||U|^3$). In addition, since the semi-quadrants are smaller than the quadrants, this optimization also reduces the average cost of anonymization.

4.5.3 Pruning Suboptimal Configurations

For any node *m* of the USeq-Btree, in the for loop of step 16, Traj-anon inspects $(d(m) - k + 1)$ configurations (all possible k-summing configurations). We realize that some of these configurations need not be considered, as they are guaranteed to be sub-optimal. In fact we claim the following lemma:

Lemma 11. For a node m with height $h(m)$ (where the height of the root is 0), any configuration in which m passes up to its ancestors the cloaking responsibility of more than $(k + 1)h(m)$ but less than $d(m)$ trajectories, is not optimal.

By Lemma 11, it suffices to compute $(k + 1)h(m)$ configurations, by simply replacing function F in step 16 of algorithm Modified Traj-anon with function $F'(m) = [0..((k + 1)h(m))] \cup \{d(m)\}$. Thus for a non-leaf node m, the algorithm computes $O(kh)$ configurations and to compute each such configuration, the "pick" action iterates over $O(kh)$ configurations of m's two children. This leads to a new upper bound of the overall running time, $O(|T_{usqb}|(kh)^3)$.

4.5.4 Precomputation

Similar to $Bulk_{dp}$, there is significant overlap in the computations across iterations of For loop in Step 16 of Modified Traj-anon. For example, if one iteration works on the M entry for (m, u), inspecting for instance (m_1, u_1) and (m_2, u_2) such that $u_1 + u_2 = u$, then the next iteration $(m, u + 1)$ will inspect the cases $(m_1, u_1 + 1)$, (m_2, u_2) and (m_1, u_1), $(m_2, u_2 + 1)$, among others. The idea is to reuse this computation across iterations.

To this end, we stage the computation in 2 parts. In the first stage we iterate over the $O(kh)$ configurations of both children to compute a temporary matrix $temp$. There are $O(kh)$ entries in this matrix and the complexity of this stage is bounded by $O((kh)^2)$. In the second stage, we create $O(kh)$ configurations using the $O(kh)$ entries of temp. Thus the running time for the second stage is also bounded by $O((kh)^2)$. Therefore the overall complexity of the modified step 16 is $O((kh)^2)$ and the overall complexity of the modified algorithm becomes $O(|T_{usqb}|(kh)^2)$.

4.5.5 Runtime Pruning

We implement a runtime optimization to further reduce the running time of the modified Traj-anon. We create the USeq-Btree top-down by successively splitting the semi-quadrants, starting from the root node. But we do not eagerly materialize all nodes of the USeq-Btree, instead, we split a (semi-)quadrant only if it contains sufficient users to maintain anonymity.

4.6 Experiments

In this section we describe a set of experiments to evaluate the effectiveness of our optimized Traj-anon algorithm. We describe experiment to show the scalability and performance of Traj-anon. We also compare the cost of anonymization and execution time of Traj-anon with those of other related anonymization techniques. Our experiments show that Traj-anon scales linearly with the number of trajectories and can anonymize up to *1 million* trajectories of length 30 within *5 min*. We show that the other anonymization techniques either have higher anonymization cost than Traj-anon (as high as *70 times*) or are extremely slow in comparison (takes more than *6 days*).

The key challenge in these experiments was to obtain user trajectory data from the location logs of commercial LBS ecosystem. To avoid any legal hurdles, we decided to use the *Brinkhoff generator* [Brib] to generate the trajectory data for our experiments. The Brinkhoff generator has been widely used to generate moving object data for studies in various fields including location-based services.

Trajectory Data. We use the Brinkhoff generator to generate the trajectories data of the mobile users in a metro area. The generator takes as input the road network of a region and generates trajectories of various classes of moving objects that are bounded by the road network. The classes differ in number and speed with which the trajectories move relative to each other. We generated a master data set of 1 million trajectories of length 30, with 5 different classes of moving objects, using the road network of the San Francisco Bay area [Brib]. Then we drew random samples of increasing number of trajectories (10k, 50k, 100k etc.) of length 10 and 30.

Platform. Unless otherwise stated all our experiments were executed using a server with 2 quad-core Intel(R) Xeon(R) processors (2.50Ghz) with 4 GB RAM and running Linux. In one of the experiment, we had to use a machine with Intel Pentium Core2 Duo processor (2.4Ghz) with 2 GB RAM and running Cygwin on Windows XP because the binary we got from the authors of [YBLW09] was compiled for that configuration.

4.6.1 Scalability

In the first set of experiments we evaluate the scalability of the optimized Traj-anon algorithm by increasing the number of trajectories to be anonymized, from 10k to 2 million, for a fixed k=50. As shown in Figure 4.11, the algorithm scales linearly with increasing number of trajectories. In fact Traj-anon can anonymize *1 million* trajectories of length 30 in less than *5 min*!

Figure 4.12 shows the break up of the time spent in the three phases of the anonymization algorithm: a) loading the user trajectories from a file to the main memory

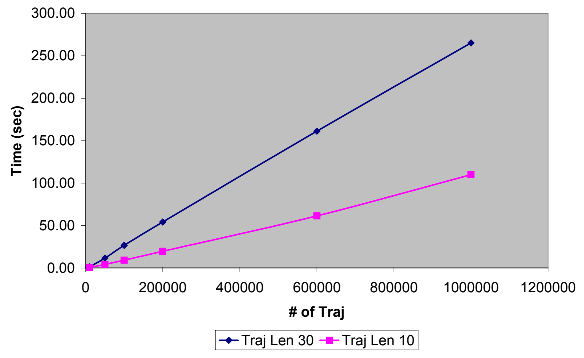


Figure 4.11: Execution time

data structures, b) obtaining the optimum configuration for the user trajectories and, c) obtaining the policy from the configuration. The anonymization step b) takes about 50%-60% of the total time while the time for obtaining the policy from the optimum configuration is less than 10%.

4.6.2 Related anonymization techniques

As described in the Section 4.7 (Related Work), the previously proposed algorithms for trajectory-aware sender k-anonymity, are not completely policy-aware and as shown in Section 3.1 the policy-aware snapshot sender k-anonymizing algorithms [DHVZ10] are not trajectory aware. Therefore as a Strawman approach we decided to extend the algorithm for policy-aware snapshot sender k-anonymity to *TP*-aware sender k-anonymity. We chose the $Bulk_{dp}$ algorithm in [DHVZ10] since it provides the optimum anonymization for a snapshot and uses quad-tree based cloaks. (since Traj-anon uses quad-cloak sequences).

In addition we consider solutions proposed for *trajectory anonymity*, a privacy problem orthogonal to sender anonymity. In trajectory anonymization, the goal is to anonymize user trajectories such that an attacker, who knows locations of users in certain snapshots (partial trajectories), cannot infer whether a user visited a particular location (i.e. whether a user's trajectory passes through a particular location). This problem

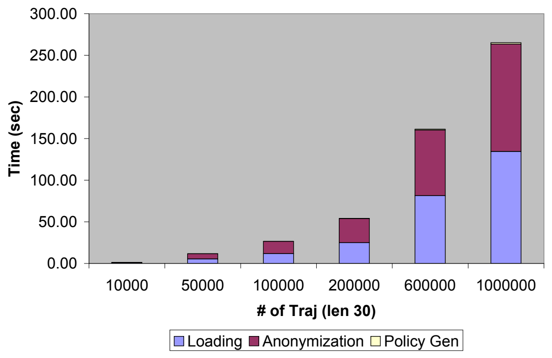


Figure 4.12: Time spent in various phases

differs from sender anonymity in two important aspects: a) the attacker does not know the complete trajectory of the user and b) minimization of cloak area can breach trajectory anonymity. We explain these in detail next.

In trajectory anonymity the assumption is that attacker does not now the complete trajectory of the users. In sender anonymity we do not make such an assumption, on the contrary, it is assumed that the attacker might know the entire trajectory of the users. Due to the difference in this assumption, some of the data transformation techniques such as *deletion of locations*, *addition of locations*, and *shifting locations* from a trajectory, that are used in trajectory anonymization solutions cannot be used in a solution for sender anonymity.

To explain the second difference, we consider a location l such that k trajectories passes through it. If a user sends a LBS request from l, then using the location (instead of a cloak that masks the location) in the sequence of cloaks used to anonymize user does not violate *TP*-aware sender k-anonymity. In fact an optimum algorithm would try to minimize the area of the cloaks as much as possible. In contrast, consider an attacker who does not know that a user trajectory passes through l. If l is used in the sequence of cloaks for anonymizing user trajectories passing through l then it violates trajectory anonymity.

Even with these differences, we identified a class of trajectory anonymization solutions whose techniques can be adapted to provider *TP*-aware sender k-anonymity. This class of solutions use some clustering algorithm to partition user trajectories into groups of k trajectories and then applies other data transformations (described earlier) to preserve trajectory anonymity. We realize that one can adapt the clustering techniques, that partitions a set of user trajectories, to provide offline *TP*-aware sender k-anonymization. We use the clustering techniques from state of the art trajectory anonymization solutions [NAS08, YBLW09] to obtain three different solutions for offline *TP*-aware sender k-anonymity. Next we describe these three solutions along with the Strawman approach based on snapshot policy-aware sender k-anonymity.

- **Strawman TP-aware** is based on the bulk anonymization algorithm $Bulk_{dp}$ described in [DHVZ10]. We format the input trajectory data as sequence of snapshots. We anonymize the first snapshot of input trajectory data using $Bulk_{dp}$ and create groups of trajectories whose locations in the first snapshot are anonymized to the same region. Since $Bulk_{dp}$ provider policy-aware sender k-anonymity each such group must have at least k members. For each group of trajectories, for each snapshot, we find the smallest quadrant that masks the locations of the trajectories in the group. Thus for each group we obtain a sequence of quadrants that we use in a bundle to anonymize the trajectories in that group. This anonymization provides *TP*-aware sender k-anonymity since there are at least k trajectories that are anonymized to the same sequence of cloaks.

- **Fast Clustering** is based on *fast TGA* in [NAS08]. It creates a cluster of k trajectories by first randomly selecting an unanonymized trajectory as the center of the cluster and then adding its k-1 nearest neighbor trajectories to the cluster. The distance between two trajectories is the sum of the "distance" between their locations in each snapshot and the distance between two locations is the log of the area of the smallest axis-parallel minimum bounding rectangle (rectangle whose sides are parallel to the x and y axis of a 2-dimensional plane) that masks the two locations.

- **Slow Clustering** is based on *multi TGA* in [NAS08]. To create a cluster of k trajectory, it first randomly selects an unanonymized trajectory as the center of the cluster and add k-1 additional trajectories one by one that minimizes the cost of the cluster. The cost of a cluster is the sum of the log of area of axis-parallel MBRs, that masks the locations of the trajectories in the cluster, in each snapshot.

- **Hilbert-based Clustering** [YBLW09] uses Hilbert index to find the nearest k-1 trajectories for each trajectory and create a cluster. The original approach requires identification of certain locations in a trajectory as quasi-identifier (uniquely identifies the user). Since we assume that entire user trajectory is accessible to the attacker, every location in his trajectory is a potential quasi-identifier. Thus in the input to the Hilbert-based clustering algorithm we specify all the locations of a trajectory as its quasi-identifier. As a result, the distance between two trajectories is the sum of the absolute difference between the Hilbert values of the locations of trajectories in each snapshot. Since the algorithm computes the clusters of k-1 nearest neighbors for each trajectory independently, two clusters can have some trajectories in common. They merge the two clusters that have one or more trajectories in common.

In the three clustering based approaches, after computing the clusters, all the trajectories in a cluster are anonymized using the sequence of axis-parallel MBRs (rectangle whose sides are parallel to the x and y axis of a 2-dimensional plane) that masks the locations of the trajectories (in the cluster) in each snapshot. We compare the execution time and cost of anonymization obtained using the four algorithms described above with those of Traj-anon. To make a fair comparison, we modify the output of Traj-anon and replace the quadrants in the cloak sequence with axis-parallel MBRs ensuring that the MBR that replaces a quadrant must be included in the quadrant.

We implement the Strawman algorithm, the fast and the slow clustering algorithms in C++ and use the executable of Hilbert-based clustering obtained from the authors of [YBLW09] that is compiled for Cygwin on Windows XP.

Figure 4.13 and Figure 4.14 shows the cost of anonymizing increasing number of trajectories (10k, 50k, 100k, 200k, 600k and 1M) and of length 10 and 30 respectively, using Traj-anon and the four anonymization algorithms described above. The x-axis in the graphs represents the number of trajectories and the y-axis represents the cost of anonymization in log scale. Figure 4.15 and Figure 4.16 shows the time it

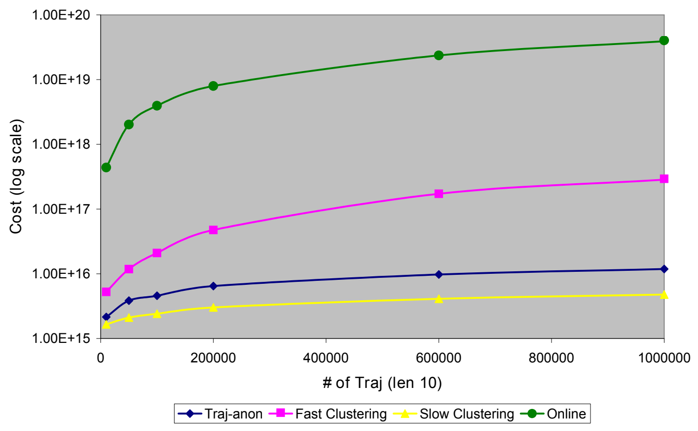


Figure 4.13: Cost: Clustering vs Traj-anon (len 10)

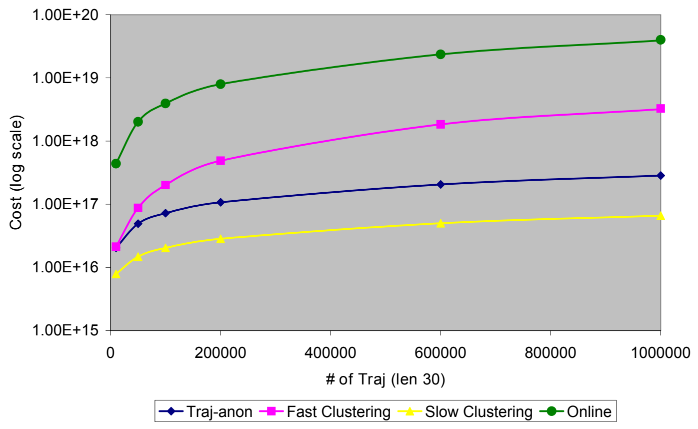


Figure 4.14: Cost: Clustering vs Traj-anon (len 30)

takes to anonymize trajectories of lengths 10 and 30 respectively, using Traj-anon and the four anonymization algorithms described above. The x-axis represents the number of trajectories and the y-axis represents the time in seconds in log scale. In all these experiments the value of k used is 50.

Comparison with snapshot-based Strawman. As shown in Figure 4.13 and Figure 4.14 the Strawman approach has the highest cost among all the anonymization algorithms. It is significantly more than than Traj-anon. In Table 4.4, it can be seen that for 600k trajectories and more the cost of anonymization with Strawman is *10 times* that of Traj-anon. This is because having optimum cost for one snapshot leads to bigger

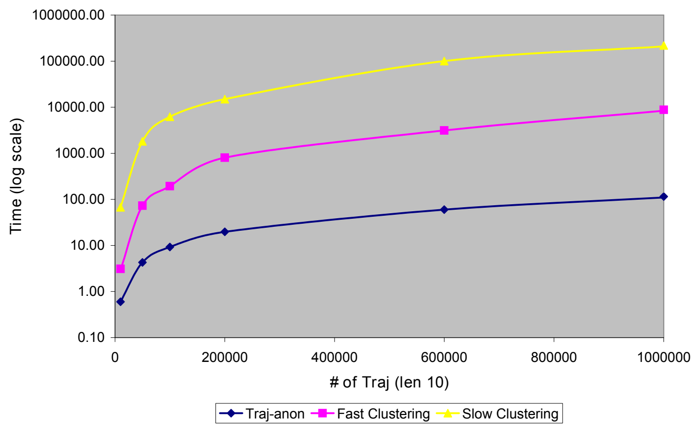


Figure 4.15: Exec time: Clustering vs Traj-anon (len 10)

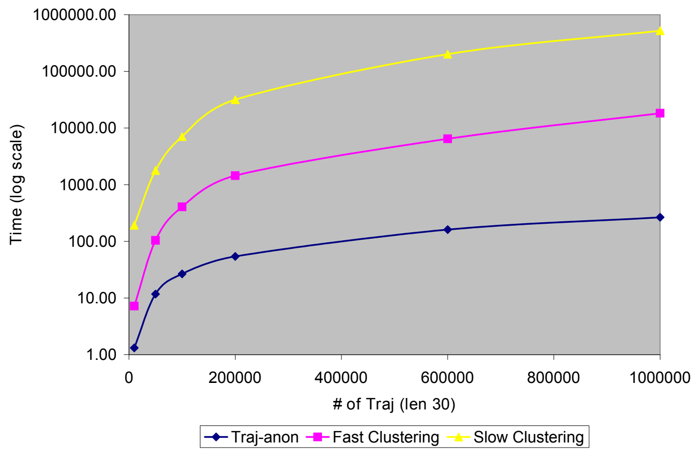


Figure 4.16: Exec time: Clustering vs Traj-anon (len 30)

cost for other snapshots when the trajectories diverge in other snapshots (since the trajectories in a group must be anonymized together in all the snapshots).

Comparison with Fast Clustering. As shown in Figure 4.13 and Figure 4.14 the cost of anonymizing trajectories using fast clustering is more than that with Traj-anon and slow clustering. As can be seen in Table 4.4, the difference between the anonymization cost increases with number of trajectories and for 1 million trajectories of length 30, the anonymization cost of fast clustering is *10 times* more than that of Traj-anon.

In terms of execution time, as shown in Figure 4.15 and Figure 4.16, the fast clustering takes a lot longer in comparison with Traj-anon. In Table 4.5 it can be seen

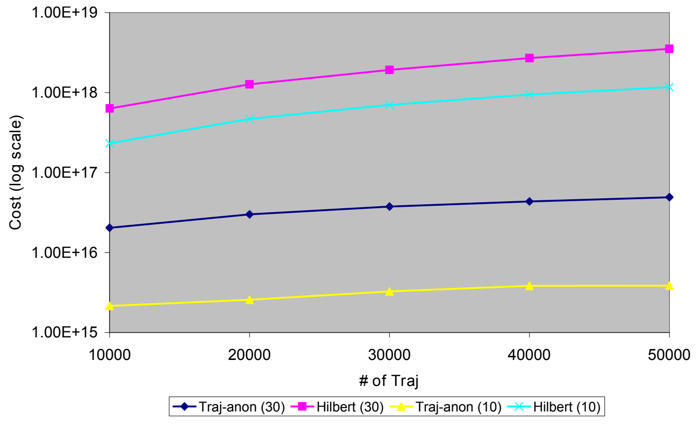


Figure 4.17: Cost: Hilbert vs Traj-anon

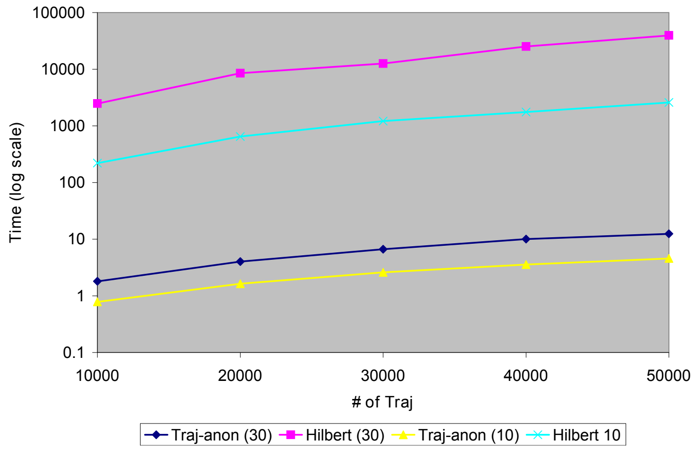


Figure 4.18: Exec time: Hilbert vs Traj-anon

that Traj-anon takes less than *5 min* to anonymize 1 million trajectories of length 30 in comparison to *302 min* by fast clustering.

Comparison with Slow Clustering. As shown in Figure 4.13 and Figure 4.14 the cost of anonymizing trajectories using slow clustering is lower than that with Traj-anon. But as shown in Figure 4.15 and Figure 4.16 it is the slowest of all the anonymization techniques. As shown in Table 4.5 it takes more than *6 days* for the slow clustering algorithm to anonymize 1 million trajectories of length 30, in comparison to *4 min* with Traj-anon. The poor performance of slow clustering is not just empirical (for this data set) but intrinsic to the algorithm due to $O(n^2)$ distance computations between n trajectories.

Comparison with Hilbert-based Clustering. Figure 4.17 compares the anonymization cost of Hilbert-based clustering with that of Traj-anon. We could not process more than 50k trajectories because the implementation of Hilbert-based clustering algorithm does not scale with number of trajectories. The cost of Hilbert-based clustering is considerably higher than Traj-anon. In Table 4.6, it can be seen that for 10k trajectories the cost of Hilbert-based clustering is *10 times* more than that of Traj-anon and the difference is *60 times* for 40k trajectories. The possible reason for the higher cost of Hilbert-based clustering is the error introduced in mapping 2-dimensional space to single dimension and using sum of Hilbert distances as the distance function.

Table 4.16 shows the running time of Hilbert-based clustering. The Hilbert-based clustering takes 7 hours to anonymize 50k trajectories of length 30, while *Traj-anon* takes 12 sec to anonymize them. Even though the Hilbert-based clustering uses a simpler distance function, since the number of comparisons to find the nearest k neighbors dominate the running time, it is slower than Traj-anon.

Table 4.4: Cost Comparison: Clustering vs Traj-anon

Algo	len	10k	50k	100k	200k	600k	1M
Straw-man	30	4.38e+17	2.02e+18	3.95e+18	7.97e+18	2.36e+19	3.92e+19
	10	6.81e+16	2.57e+17	4.66e+17	9.68e+17	2.76e+18	4.53e+18
Fast Clust	30	2.12e+16	8.74e+16	2.02e+17	4.89e+17	1.84e+18	3.20e+18
	10	5.25e+15	1.18e+16	2.09e+16	4.75e+16	1.72e+17	2.83e+17
T-anon (MBR)	30	2.04e+16	4.91e+16	7.22e+16	1.07e+17	2.06e+17	2.85e+17
	10	2.15e+15	3.85e+15	4.57e+15	6.47e+15	9.76e+15	1.18e+16
Slow Clust	30	7.87e+15	1.48e+16	2.03e+16	2.84e+16	5.00e+16	6.59e+16
	10	1.65e+15	2.10e+15	2.41e+15	3.02e+15	4.11e+15	4.77e+15

Table 4.5: Running Time Comparison: Clustering vs Traj-anon

Algo	len	10k	50k	100k	200k	600k	1M
T-anon (MBR)	30	1.32	11.72	26.69	54.20	161.24	265.22
	10	0.60	4.30	9.27	19.77	63.78	110.00
Fast Clust	30	7.17	104.04	407.52	1444.60	6466.43	18176.67
	10	2.96	73.12	193.71	805.99	3130.31	8391.40
Slow Clust	30	192.27	1781.35	7075.29	31664.37	201490.11	521191.83
	10	67.35	1818.98	6205.36	14972.30	100617.18	209226.44

Table 4.6: Cost Comparison: Hilbert vs Traj-anon

Algorithm	len	10k	20k	30k	40k	50k
Traj-anon (MBR)	30	2.04e+16	3.00e+16	3.76e+16	4.36e+16	4.91e+16
	10	2.15e+15	2.57e+15	3.27e+15	3.84e+15	3.85e+15
Hilbert Clustering	30	6.32e+17	1.27e+18	1.92e+18	2.70e+18	n/a
	10	2.32e+17	4.69e+17	7.02e+17	9.47e+17	1.17e+18

Table 4.7: Running Time Comparison: Hilbert vs Traj-anon

Algorithm	length	10k	20k	30k	40k	50k
Traj-anon (MBR)	30	1.8	4.03	6.66	10	12.41
	10	0.78	1.64	2.6	3.56	4.56
Hilbert Clustering	30	2478	8522	12587	25206	
	10	220	650	1216	1758	2584

4.7 Related Work

In the context of LBS, the two aspects of privacy that have received most attention are *trajectory anonymity* and *sender anonymity*.

Trajectory privacy. As mentioned earlier, the line of work on *trajectory privacy* [NAS08, YBLW09, TM08, MKA$^+$08] is complementary to this paper, as trajectory privacy refers to hiding user's precise location over a period of time (one is not required to hide the identity of the user) while sender anonymity refers to hiding the identity of the user where one is not required to hide the trajectory, on the contrary, one assumes it falls in the attacker's hand. The problem earlier referd to as *location privacy* can be viewed as a special case of trajectory privacy where trajectory length is 1.

Even though the problem of trajectory privacy is orthogonal to the problem studied in this paper, as described in Section 3.6 a class of clustering based solutions can be adapted to provide offline policy-aware trajectory-aware sender k-anonymity. Not all the techniques used for trajectory anonymity such as *Point Suppression* [TM08] or *Trajectory Suppression* [NAS08] or *Space Translation* [ABN08] or *Randomized Reconstruction* [NAS08] are applicable for sender anonymity since they assume that the actual trajectories of the users are not available to the attacker.

Classes of Attackers. The solutions for sender k-anonymity in the context of location-based services can be classified into 4 categories based on the class of attackers they prevent against:

- Policy-unaware trajectory-unaware: The solutions [GG03, MCA06, KGMP07] in this class are also known as *k-inside* policies [DHVZ10] as these solutions use tightest cloak (of a pre-defined shape) that includes the sender and k-1 other users. This class of solutions neither preserve privacy against a policy-aware (as shown in [DHVZ10]) nor against a trajectory-aware attacker (also shown in [BWJ05, XC07, CM07]).
- Policy-aware trajectory-unaware: This class of solutions [BMWJ07, DHVZ10] ensure that there are at least k users anonymized using the same cloak. The privacy guarantee of these solutions is strictly stonger than the policy-unaware solutons i.e. they also defends against the policy-unaware attacker but not vice-versa. But as shown in Section 3.1 they fail to preserve privacy against a policy-aware trajectory-aware attacker.
- Policy-unaware trajectory-aware: This class of solutions [XC07, GDVM09, CM07, BWJ05] tries to provide anonymity against the trajectory-aware attackers using a sequence of cloaks that masks the user and the same k-1 users for the entire duration of the user trajectory. The claims of policy-awareness in [CM07] are debatable as it is not clear what is known to the policy-aware attacker: a) the function that maps the users trajectories to the sequence of cloaks or b) or the algorithm by which the function is obtained in addition to the function itself. This argument is illustrated by showing the privacy problem with a *2-sharing* policy in the Example 22 below.

  Example 22. Consider the cloaking algorithm in [CM07] that takes into account the requesting location to generate cloaking *groups* (set of locations that are cloaked to the same region). For locations in Figure 4.19, if the first request is made by C the algorithm groups C with B whereas if the first request is made by B then it puts B and A in the same cloaking group to satisfy 2-sharing property. In the case when the initial request contains the cloak corresponding to $\{C, B\}$, a policy-aware attacker can infer that the sender is C!

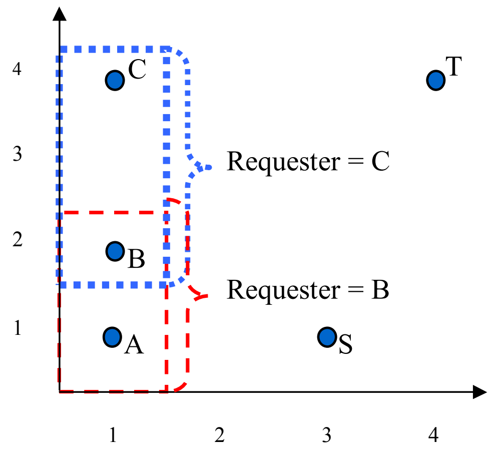


Figure 4.19: 2-sharing policy

- Policy-aware trajectory-aware: We are not aware of any work that provides complete policy-aware trajectory-aware sender k-anonymity and therefore in this paper we propose a solution to provide policy-aware trajectory-aware sender k-anonymity. As illustrated in the Section 3.1 even this privacy guarantee does not allow to completely publish the linkage between multiple requests sent by the same user. It does allow to publish the requests made along a (set of) trajectory that is very useful for researchers, advertisers and wireless network management tasks.

LBS:Trusted or Untrusted. Most of the sender anonymity solutions [GG03, MCA06, KGMP07, BWJ05, DHVZ10, BMWJ07, XC07, GDVM09, CM07] till date assume that users need not trust the LBS providers and the LBS requests should be anonymized before they are sent to the LBS. We are not aware of the use of any of these anonymization methods by the users of LBS, hence in the model used in this paper we assume that the LBS provider is a trusted entity and responsible for anonymizing the user requests that it collects over a period of time. We share this assumption with a line of work on trajectory anonymization [NAS08, YBLW09, TM08, ABN08, MFD09, MKA$^+$08] where the location provider (who logs user trajectory) is trusted and is responsible for anonymizing the trajectory data.

Online vs Offline. Another contrasting feature between the previous trajectory-aware sender anonymity proposals and that in this paper is the *mode* of anonymization. In [XC07, GDVM09, CM07, BWJ05] LBS requests are anonymize as they are issued i.e. *Online* while we anonymize the requests after there response has been sent i.e. *Offline*. One can possibly use the online solutions for the offline trajectory-aware policy-aware sender anonymity problem but with a greater cost since the future movement of the users are not known at the time of anonymization and the cloak that masks the same set of k users can become arbitrarily large if their trajectories diverge.

4.8 Conclusions

We introduce the problem of offline trajectory-aware policy-aware sender k-anonymity. We show that trajectory-awareness requires stronger defense and extend the snapshot policy-aware sender k-anonymity to trajectory-aware attackers. We show that trajectory-aware policy-aware anonymization is harder than snapshot policy-aware anonymization and in particular finding optimum trajectory-aware policy-aware anonymization is NP-hard. We propose a polynomial time l-approximation algorithm for anonymizing trajectories of length l and empirically show the effectiveness of this algorithm.

4.9 Proofs

4.9.1 Lemma 6

Proof. **(a)** Let U be a set of n trajectories and policies P_1 and P_2 are equivalent for anonymizing U w.r.t. a G-tree T. We describe the cost of anonymizing U using P_1 as:

$$Cost(P_1, U) = Cost(m_1) + Cost(m_2) + \ldots + Cost(m_n)$$

where $m_i \in T$ and m_i = $P_1(U, u_i)$ for $1 \leq i \leq n$. Note, for $i \neq j$, m_i and m_j can be the same node in T. Similarly we describe the cost of anonymizing U using P_2 as:

$$Cost(P_2, U) = Cost(m'_1) + Cost(m'_2) + \ldots + Cost(m'_n)$$

where $m'_i \in T$ and m'_i = $P_2(U, u_i)$ for $1 \leq i \leq n$. Note, for $i \neq j$, m'_i and m'_j can be the same node in T. Since P_1 and P_2 are equivalent, if a node $m \in T$ is used by P_1 to anonymize x trajectories in U then m is also used by P_2 to anonymize the same number of trajectories in U. Therefore,

$$\begin{multline} Cost(m_1) + Cost(m_2) + \ldots + Cost(m_n) \\ = Cost(m'_1) + Cost(m'_2) + \ldots + Cost(m'_n) \quad (4.1) \end{multline}$$

because each quadrant appears same number of times on both sides of the Equation 4.1. Therefore,

$$Cost(P_1, U) = Cost(P_2, U)$$

(b) Let P_1 provides TP-aware sender k-anonymity to U w.r.t. T. Therefore, for each node $m \in T$, P_1 either anonymizes $(\geq k)$ trajectories to m or $none$. We are given that P_1 and P_2 are equivalent for T, therefore they both anonymize the same number of trajectories using m. Therefore, P_2 either anonymizes $(\geq k)$ trajectories using m or $none$. Thus, P_2 also provides TP-aware sender k-anonymity to U w.r.t. T. Similarly we can show that if P_2 provides TP-aware sender k-aonymity to U, so does P_1.

Next we assume that P_1 does not provide TP-aware sender k-anonymity to U. Hence, there must exist a node $m \in T$ that is used by P_1 to anonymize $1 \leq i < k$ trajectories. Since P_1 and P_2 are equivalent, P_2 also anonymizes i trajectories using the node m. Since $1 \leq i < k$, P_2 does not provide TP-aware sender k-anonymity to U. $\square$

4.9.2 Lemma 7

Proof. We describe the cost of anonymizing the set U of n trajectories using policy P as:

$$\begin{aligned} Cost(P, U) &= \sum_{u \in U} Cost(P(U, u)) \\ &= Cost(m_1) + Cost(m_2) + \ldots + Cost(m_n) \end{aligned} \quad (4.2)$$

where $m_i \in T$ and m_i = $P(U, u_i)$ for $1 \leq i \leq n$. Since, for $i \neq j$, m_i and m_j can be the same node in T, we can rewrite the above equation as follows:

$$Cost(P, U) = \sum_{m \in T} f'(m, P) \times Cost(m)$$

where $f'(m, P)$ is the number of trajectories anonymized by P using cloak sequence m. Since C represents the equivalence class of P,

$$\forall m \in T, \quad f'(m, P) = f(m, C)$$

where $f(m, C)$ is as defined in Definition 17. Therefore the cost of configuration C of T can be written as:

$$\begin{aligned} Cost_c(C, U) &= \sum_{m \in T} f(m, C) \times Cost(m) \\ &= \sum_{m \in T} f'(m, P) \times Cost(m) \\ &= Cost(P, U) \end{aligned} \tag{4.3}$$

□

4.9.3 Lemma 8

Proof. Let U be a set of trajectories and T be a G-tree of quad-cloak sequences. Let P be a policy that uses the cloak sequences from T and C be the configuration representing the class of policies equivalent to P.

First we assume that P provides ***TP***-aware sender k-anonymity and show that C is $k - summing$ configuration. Since P is ***TP***-aware sender k-anonymous, each quad-cloak sequence in T is used in P to anonymize either $\geq k$ trajectories or $none$. Thus

- for a leaf node $m \in T$
  - (i) If $d(m) < k$, then P cannot anonymize any trajectory using m, therefore $C(m) = d(m)$.
  - (ii) if $d(m) \geq k$, then P could either anonymize $\geq k$ trajectories or $none$. In former case $C(m) \leq (d(m) - k)$ while in later case $C(m) = d(m)$.
- for an internal node $m \in T$ let $\Delta = \sum_{i=1}^{l} C(m_i)$,
  where $m_1 \dots m_l$ are the children of m in T
  - (iii) if $\Delta < k$ then there are total ¡ k trajectories passed up by children of m. Thus P cannot anonymize any trajectory using m and therefore, $C(m) = \Delta$.

(iv) if $\Delta \geq k$ then the children of m passes up $\geq k$ trajectories. Therefore, P could either anonymize $\geq k$ trajectories or $none$. In former case $C(m) \leq (\Delta - k)$ while in later case $C(m) = \Delta$.

Thus C satisfies $k - summing$ property.

Next we assume that C is k-summing configuration and show that P provides ***TP***-aware sender k-anonymity to U. Equivalently we show that under C, each cloak of T is used to anonymize either $\geq k$ trajectories or $none$. Since C is k-summing configuration, it implies:

- for a leaf node $m \in T$

  (i) if $d(m) < k$, then $C(m) = d(m)$. Thus P does not anonymize any trajectory using m.

  (ii) if $d(m) \geq k$, then either $C(m) = d(m)$ or
  $C(m) \leq (d(m) - k)$. In the later case P anonymizes $\geq k$ trajectory using m, while in former case $none$.

- for an internal node $m \in T$ let $\Delta = \sum_{i=1}^{l} C(m_i)$,
  where $m_1 \ldots m_l$ are the children of m in T

  (iii) if $\Delta < k$, then $C(m) = \Delta$. Thus P does not anonymize any trajectory using m.

  (iv) if $\Delta \geq k$, then either $C(m) = \Delta$ or $C(m) \leq (\Delta - k)$. In the later case P anonymizes $\geq k$ trajectories using m, while in former case $none$.

Thus P provides ***TP***-aware sender k-anonymity.

□

4.9.4 Lemma 9

Proof. We use structural induction to prove that each node m in the G-tree T and an integer $l \leq d(m)$,

$$cost_{alg}(m, l) = cost_{min}(cset(m, l))$$

, where $cost_{alg}(m, l)$ represents the cost computed by Traj-anon for passing up l (unanonymized) trajectories at m and $cost_{min}(cset(m, l))$ represents minimum cost of passing up l trajectories at m among all the k-summing configurations that passes up l (unanonymized) trajectories at node m.

Basis: For a leaf node m and an integer $l \leq d(m)$, it is obvious by construction that $cost_{alg}(m, l) = cost_{min}(cset(m, l))$.

Induction: Let m be a non-leaf node in the G-tree T and $m_1, m_2, \cdots m_f$ be the children of m. Let l be an integer such that $l \leq d(m)$ and $cset(m, l)$ be the set of k-summing configuration that passes up l (unanonymized) trajectories at node m. We show that for each configuration $g \in cset(m, l)$, $cost_{alg}(m, l) \leq cost(g(m))$, where $cost(g(m))$ represents the cost of g at node m. If $g(m_1) = l_1$, $g(m_2) = l_2$, $\cdots$ and $g(m_f) = l_f$, the cost of m in g can be written as

$$\begin{aligned} cost(g(m)) := [&cost(g(m_1)) + cost(g(m_2)) + \cdots + cost(g(m_f)) \\ &+ cost(m) \times (l_1 + l_2 + \cdots + l_f - l)] \end{aligned}$$

By induction hypothesis we assume that $cost_{alg}(m_1, l_1) \leq cost(g(m_1))$, and similarly $cost_{alg}(m_2, l_2) \leq cost(g(m_2))$, $\cdots$, $cost_{alg}(m_i, l_i) \leq cost(g(m_i))$, $\cdots$, $cost_{alg}(m_f, l_f) \leq cost(g(m_f))$. Therefore

$$\begin{aligned} &cost_{alg}(m_1, l_1) + cost_{alg}(m_2, l_2) + \cdots + cost_{alg}(m_f, l_f) \\ &\leq cost(g(m_1)) + cost(g(m_2)) + \cdots + cost(g(m_f)) \end{aligned}$$

And by adding the constant value $cost(m) \times (l_1 + l_2 + \cdots + l_f - l)$ to both the sides we get

$$cost_{alg}(m, l) \leq cost(g(m))$$

Similarly, for each node m and each integer $l \leq d(m)$, and each configuration $g \in cset(m, l)$, we can show that $cost_{alg}(m, l) \leq cost(g(m))$. Therefore $cost_{alg}(m, l) = cost_{min}(cset(m, l))$. □

4.9.5 Lemma 10

Proof. Let the optimum k-summating configuration for T_u be C. Since all the nodes in T_u are also in T_{usq} we can define a configuration C' for T_{usq} as follows

- $C'(m) = C(m)$ for $m \in T_{usq}$ and $m \in T_u$
- $C'(m) = \sum_{l=1}^{4} C'(m_i)$ for $m, m_1 \ldots m_4 \in T_{usq}$ and $m \notin T_u$, where $m_1 \ldots m_4$ are child nodes of m

The above conditions ensure that C' only uses those nodes of T_{usq} for anonymization that are also in T_u. Each node m that is not in T_u pass-up all the trajectories that are passed up by m's child nodes to be anonymized by m's ancestors. C' is a valid configuration since $\forall m$, $C'(m) < d(m)$ and $C'(m) \leq \sum_{l=1}^{4} C'(m_i)$. C' is k-summing since C is k-summing and $Cost(C')$ = $Cost(C)$ since the newly inserted nodes are not used for anonymization and the nodes that are used for anonymization have the same cost in the two G-trees. □

4.9.6 Lemma 11

Proof. Let U be a set of user trajectories, B a USeq-Btree, C an optimal configuration of B, and P a policy it represents. Suppose there is a node (cloak sequence of semi-quadrants) $m \in B$ such that $(k+1)h(m) < C(m) < d(m)$. Then there is a set S of at least k trajectories such that (i) all the trajectories in S are masked by m, and (ii) each trajectory in S is anonymized by P using some ancestor a of m, and (iii) if all the trajectories in S are removed, the cloak sequence they are mapped to under P, continue to anonymize at least k trajectories. We construct a policy P' that anonymize the trajectories in S using m instead of its ancestors. P' continues to be ***TP***-aware sender k-anonymous, but has lower cost, contradicting the optimality of P. □

4.9.7 Theorem 3

Proof. We prove this by reducing the decision version of optimum policy-aware snapshot k-anonymization with circular cloaks to the decision version optimum offline ***TP***-aware k-anonymization with circular cloaks. First we briefly describe the problem of policy-aware snapshot k-anonymization with circular cloaks.

Let D be an instance of location database with schema $S = \{userid, locx, locy\}$ and SC be a set of points in 2-dimensional space. A snapshot policy with circular cloaks is defined as a deterministic function that maps locations in D to circular cloaks, each centered at some point from SC, with no restriction on radius. The cost of a snapshot policy with circular cloaks is computed as:

$$Cost_s(P, D) = \sum_{l \in D} Cost(P(D, l))$$

where the cost of the cloak $P(D, l)$ is the area of circular cloak.

Definition 19 (Snapshot k-anonymity with circular cloaks)**.** Given an instance D of location database and SR be the set of points in 2-dimensional space. Is there a snapshot policy P with circular cloaks that provides policy-aware sender k-anonymity and whose $Cost_s(P, D) \leq C$.

We reduce an instance I of the above problem to an instance I' of the optimum offline ***TP***-aware sender k-anonymity with circular cloaks. For each tuple $t \in D$, we create an user-history object u with trajectory of length 1 and set $u.userid() = t.userid$ and $u.location(1) = (t.locx, t.locy)$. Let the resulting set of user-history objects be U. We create an instance I' of optimum offline TP-aware k-anonymization with circular cloaks using U and SR. We prove that there is snapshot policy P with circular cloaks that provides policy-aware snapshot sender k-anonymity w.r.t. D and $Cost_s(P, D) \leq C$, if and only if there is an anonymization policy P_t that provides TP-aware sender k-anonymity solution w.r.t. U and $Cost(P_t, U) \leq C$.

Let P_t be an offline policy that uses circular cloaks and provides ***TP***-aware sender k-anonymity to the set U of user-history objects (constructed above). Let the cost of P_t be $Cost(P_t, U) \leq U$. Since the user-objects in U are of length 1, the bundles obtained with P_t are also of length 1. We use P_t to obtain a snapshot policy P_s for D as follows. For each tuple $t \in D$, we define $P_s(D, (t.locx, t.locy)) = b.cloak(1)$ where $b = P_t(U, u)$ for the user-history object u such that $u.userid() = t.userid$. Since P_t provides ***TP***-aware sender k-anonymity, there exists at least k user-history objects that are anonymized to a bundle b. Therefore the policy P_s as defined above also anonymizes at least k locations to the circular cloak in bundle b. Hence P_s provides policy-aware snapshot k-anonymity. Moreover, since $Cost(P_t, U) \leq C$ and $Cost(P_t, U) = Cost_s(P_s, D)$, therefore $Cost_s(P_s, D) \leq C$.

Suppose there exists an snapshot policy P_s with circular cloaks that provides policy-aware sender k-anonymity to D and whose cost $Cost_s(P_s, D) \leq C$. We use P_s to obtain a policy P_t as follows. For every user-history object $u \in U$ corresponding to the tuple $t \in D$ such that $u.userid() = t.userid$, we define $P_t(U, u) = b$ where b is a bundle of length 1 and $b.cloak(1) = P_s(D, (t.locx, t.locy))$. Since P_s provides policy-aware snapshot sender k-anonymity, there exists at least k locations that are aonymized to the same cloak. Therefore there are at least k user-history objects that are anonymized to the same bundle and P_t is ***TP***-aware sender k-anonymous. Moreover, since the $Cost_s(P_s, D) \leq C$, and $Cost(P_t, U) = Cost_s(P_s, D)$, therefore $Cost(P_t, U) \leq C$. $\square$

4.9.8 Theorem 4

Proof. We prove this by reducing the problem of *optimum* $k - anonymization$ *of relational tables* on binary alphabet, shown to be NP-hard in [PBD09, BW10], to the optimum offline ***TP***-aware sender k-anonymity with quad-cloaks.

We first briefly describe the problem of optimum $k - anonymization$ of relational tables on binary alphabet with suppression. Let T be a relational table with m columns where each tuple contains data corresponding to a unique user. The tuples of T can be considered to be m-dimensional vectors v_i drawn from Σ^m, where $\Sigma = \{0, 1\}$. Thus T can be also be represented as a subset $T \subseteq \Sigma^m$. Let $\star$ be a fresh symbol not in Σ. We define suppression as follows:

Definition 20 (Suppressor)**.** Let f be a map from T to $(\Sigma \cup \{\star\})^m$. We say f is a a *suppressor* of T if for all $t \in T$ and $j = 1, \ldots, m$ it is the case that $f(t)[j] \in \{t[j], \star\}$.

Inuitively, a suppressor function replaces the values of certain attributes in certain tuples with $*$. The idea behind a suppressor function is that by replacing values of certain attributes in a set of tuples with $*$, it can make all of them identical such that an attacker cannot associate a tuple in that set to the actual user. This is formalized in the following definition.

Definition 21 (k-Anonymity)**.** Let T be a relational table and f a suppressor function. The anonymized table $f(T)$ is said to be k-anonymous if for any $t \in T$, there exist k distinct vectors in T such that $f(t_1) = f(t_2) = \cdots = f(t_k) = f(t)$.

Since there can be many possible suppressor functions, the goal is find the one that minimizes the number of suppressed values i.e. number of $*$ in the anonymized table. We define the problem of optimum k-anonymization of relational tables on binary alphabets as follows.

Definition 22 (optimum k-anonymity)**.** Given a relational table $T \subseteq \Sigma^m$, and a positive integer $c \in \mathbb{N}$, is there a suppressor f such that $f(T)$ is k-anonymous, and the total number of vector coordinates suppressed in $f(T)$ is at most c?

For ease of presentation we use k=3 and reduce the optimum 3-anonymity of relational table on binary alphabet to optimum offline *TP*-aware sender 3-anonymity. Given a relational table T with n m-dimensional tuples over binary alphabet $\{0, 1\}$, we create a set of n user-history objects with trajectories of length m. For each tuple t_i, we create a user-history object u_i and set the location at the j_{th} snapshot of the trajectory as

- (i, i) if the j-th column in t_i has the value 0.
- $(nr + i, i)$ otherwise, where $r = 2^{\lceil \lg \sqrt{2mn} \rceil}$.

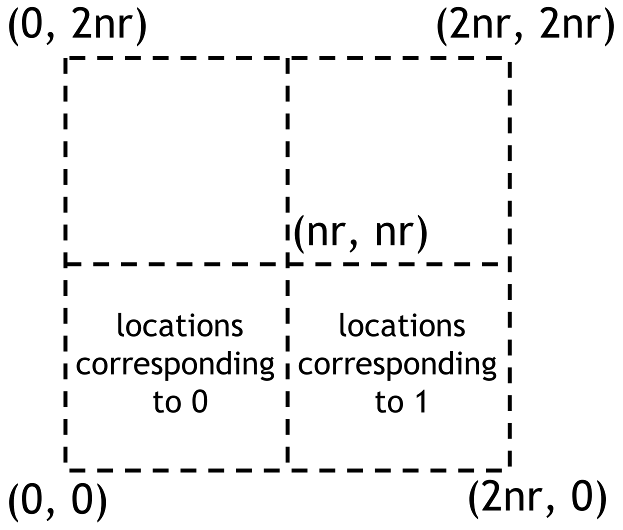


Figure 4.20: Locations corresponding to the binary data in a Relational Table

We construct a quad-tree Q such that the root node represents the square region $(0, 0)$ (left-bottom coordinates) to $(2nr, 2nr)$ (right-top coordinates) as shown in Figure 4.20. The root quadrant is divided into 4 equal square sub-quadrants. We show that the cost of the optimal 3-ANONYMITY solution for T is at most c if and only if the cost of the optimum policy that provides *TP*-aware sender k-anonymity to the set of users U constructed above is at most $(4c + 1)n^2r^2$.

Suppose that there is a solution that finds the optimum quad-cloak policy P of cost at most $(4c+1)n^2r^2$. We construct a suppressor f that k-anonymizes T as follows. For any $1 \leq i \leq n$ and any $1 \leq j \leq m$, if the j_{th} location in the trajectory of u_i is masked by the root node of Q in the cloak sequence used to anonymize u_i, then $f(t_i)[j] = \star$ and $f(t_i)[j] = t_i[j]$ otherwise. Given the upper bound on the cost of the policy there can be at most c such locations in the trajectories of the users objects that are to the root node of Q in the cloak sequences used to anonymize them. Therefore the cost of f is at most c. Moreover, since P preserves sender 3-anonymity, there must be 3 trajectories that are anonymized to the same cloak sequence and by construction these 3 trajectories will be anonymized the same way by the suppressor f and hence f is 3-anonymous.

Next let assume f is a suppressor that provides 3-anonymity to T and whose cost is at most c. Using f we define a quad-cloak policy P for the set U of user-history objects constructed above. If the value in the j_{th} column in the i_{th} tuple of T is not suppressed by f, then

- If the value is 0, then P uses a cloak sequence with the quadrant $(0, 0)$ to (n, n) at the j_{th} position to anonymize user u_i.
- If the value is 1, then P uses a cloak sequence with the quadrant $(nr, 0)$ to $(nr + n, n)$ at the j_{th} position to anonymize user u_i.

P is a valid policy as every cloak used masks the corresponding location. For any two tuples t_a and t_b, $f(t_a) = f(t_b)$ implies that P uses the same cloak sequence to anonymize users-history objects u_a and u_b. Given that f provides 3-anonymity to T there must be 3 users that are anonymized using the same cloak sequence hence P provides *TP*-aware 3-anonymity to the set of users U. Furthermore, since the cost of f is at most c, there are at most c suppressions and hence at most c locations in the trajectories of the users in U are anonymized by P to the root node of Q. The sum of the area of these cloaks is at most $4cn^2r^2$. The remaining locations in the trajectories of users of U are anonymized using cloaks of size n^2. Since there are at most mn such locations, the total cost of P is $4cn^2r^2 + mn^3 \leq 4cn^2r^2 + 2mn^3 \leq (4c+1)n^2r^2$. $\square$

4.9.9 Theorem 5

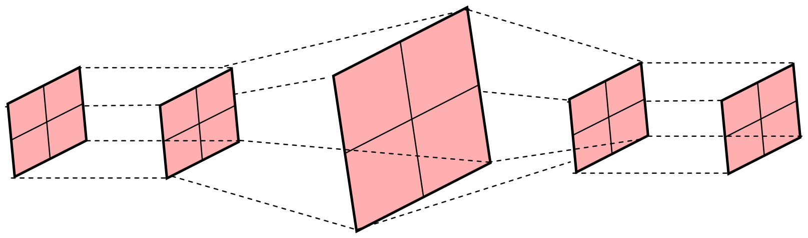


Figure 4.21: Optimum cloak sequence

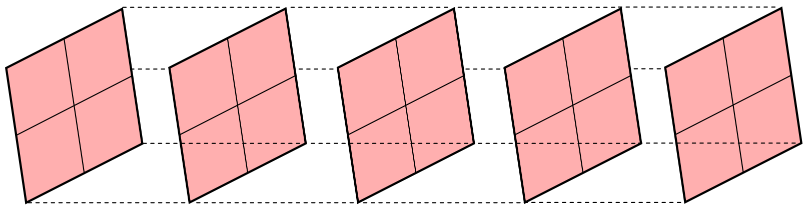


Figure 4.22: Uniform cloak sequence

Proof. Let policy P_{opt} be the optimum quad-cloak policy that provides *TP*-aware sender k-anonymity to U using cloaks from Q. Let u be a trajectory in U that is anonymized by P_{opt} using a cloak sequence $s = \{q_1, q_2, \ldots, q_l\}$. Let R be the size of the biggest cloak in s and let the area of all the other cloaks in s is very small i.e. ϵ, therefore the cost of s is $R + (l-1) \times \epsilon$. Given a quad-cloak $x \in Q$ whose size is less than or equal to R, we denote by $f_R(x)$ the quad-cloak of size R that contains x in Q, i.e. $f_R(x)$ is either x itself or an ancestor of x. We construct P'_u such that $P'_u(u)$ uses the sequence of cloaks $s' = \{f_R(q_1), f_R(q_2), \ldots, f_R(q_l)\}$. Clearly the cost of sequence s' is $l \times R$ that is less than $l \times cost(s)$. Therefore the cost of P'_u is at most l times the cost of P_{opt}.

Since P_{opt} is *TP*-aware sender k-anonymous, there are at least k trajectories anonymized using cloak sequence s. By construction the same k or more users are anonymized using cloak sequence sequence s'. Therefore the policy P'_u as constructed above provides *TP*-aware sender k-anonymity. □

4.9.10 Theorem 6

Proof. Let U be a set of trajectories. Lemma 9 shows that **Traj-anon** computes optimum uniform quad-tree policy for anonymizing U. According to Theorem 5 this optimum solution is l-approximation of the optimum quad-tree policy for anonymizing U. Therefore, **Traj-anon** computes the l-approximation solution to the problem of optimum *TP*-aware sender k-anonymity using quad cloaks. □

4.10 Acknowledgements

My co-author and collaborator, Kevin Keliang Zhao at UCSD and Richard Hull at IBM, together with Alin Deutsch, made important contributions to the material that formed the basis of this chapter.

Chapter 5

Legal Requirements and Industrial Practices

5.1 LBS Privacy Laws

Privacy is a sensitive issue in the modern democratic society and is often considered a fundamental right of an individual. Its scope and expectation often depends upon the social and political culture of the society and thus varies across national and regional boundaries. In many countries there are legal requirements (applicable even for the government agencies) on honoring the privacy of individuals. In this section we look at the privacy laws of two regions, United States and European Union, that relates to the collection and use of location data and LBS requests.

5.1.1 US Privacy Laws for Location-based Services

In United States the right to privacy is acknowledged very strongly in the constitution itself. In the context of LBS, its important to consider the possible application of the Fourth amendment.

Constitutional protection

The Fourth amendment protection states the requirement of a warrant and a probable cause prior to any search or seizure. Whether the consumer location falls under the umbrella protection of Fourth amendment has been debated in the courts and verdicts favoring both sides of the argument have been seen in the recent past. A couple of supreme court rulings, something known as "third-party doctrine", states that once information is revealed to third-party businesses, there should be no reasonable expectation of privacy. But, this doctrine was established by the supreme court in the pre-Internet era and in recent times the Fourth amendment protection has been extended to the contents of documents, files and networked computers accessible to third-parties. But the Fourth amendment protection only applies to federal and other government agencies. It was stated in a recent hearing [hea] of Senate Judiciary Subcommittee on Privacy, Technology and the Law that as of now the Fourth amendment protection doesn't apply to businesses.

Outdated Statutory Laws

The Electronic Communications Privacy Act (ECPA) is a statutory federal law introduced in 1980s to protect electronic communications and records (e.g. files) against possible wrongful use and public disclosure by law enforcement agencies or unauthorized private parties. But the statutory law was introduced in 1986 and have not been updated to addressed the privacy issues associated with most of the new communication technologies that are in use today. The ECPA dictates separate privacy requirements for a "communication" service from a "remote computing" service. With the LBS, its difficult to adjudicate whether a Location-based service constitutes a communication service or a remote computing service. This has resulted in following inconsistent treatment of the LBS data (location and LBS request).

For e.g. when a person uses a smartphone to place a phone call, the wireless communication service provider cannot disclose his location (at that instant) to a third party without his explicit consent. But when the same person uses the same smartphone and accesses a LBS (to find a nearest restaurant) then the wireless communication service provider is legally free to disclose his location to anyone without the explicit or implicit consent of the person.

Location Privacy Act of 2011

United state legislature has introduced a new bill named *Location Privacy Act of 2011* that would require a company to get a customer's consent before collecting his location data and before sharing that with third parties. The bill is mainly focused on avoiding the use of location data for unlawful purposes such as stalking while allowing the companies to obtain this data for innovative services.

CTIA Recommendations

In the absence of a clear Law addressing the issue of privacy in LBS, CTIA, the International Association for the Wireless Telecommunications industry, has proposed a set of best practices [cti] for LBS providers. The most important element of these recommendations is *notice*. It states that the LBS provider should give explicit notice to the users as to how their location information is used by the LBS provider. It should also disclose its data retention policy (e.g. data retention period) and the measures taken by the LBS provider to safeguard this data. This should be done irrespective of whether the data is retained or used in its original form or aggregated or anonymized form.

Shortcomings

The Law in United States is unclear on the privacy expectations of a LBS user regarding his location data and LBS requests. It is not clear to the users what kind of data is collected by the various entities (LBS providers, Wireless communication service providers, Location Providers, Smartphone manufacturers) and what methods are used to protect user's interest while the data is under their control. Its also unclear for the LBS providers, Location servers and other entities that collect the data, when they should and when they should not share this data with third parties or law enforcement agencies.

The bill proposed to address the privacy concerns of the LBS users, is focused mainly on obtaining user's consent before collecting and sharing his data. It does not address the issue of protecting user's interest once he gives the consent for collecting and using his location and LBS requests. More specifically there are no requirements stating the duration for which such data can be kept, or on how it should be anonymized or aggregated.

It also does not address the issue when the subscriber of a wireless service is different from the user (e.g. the family plan). In many cases, the actual user may not want the subscriber to be the authority for the consent for collecting and sharing the location and LBS requests (e.g. the spouse may not want the other spouse to be the authority for his/her location and LBS requests). In the current scenario the wireless communication service provider only knows about the subscriber and not the user. Hence, the proposed law needs to address this scenario and ensure that the consent of the actual user is required.

It should be realized that user's consent is not required for every instance the data is collected, used or shared. The "once for all" kind of consent makes it difficult for the users to realize when and how his data is collected, used or shared. Moreover to avoid user anxiety and scrutiny, such a consent may be included in the middle of a lengthy service agreement, which is a common practice for obtaining user consent in non-LBS services and products. Or the LBS provider (and other entities) can make it such that if a user wants to use a LBS service, he has to give the consent or not use the service. It should be realized that in some ways this is even worse than not having the law since it gives a false sense of privacy to the users, and provides legal validation and ethical justification to the entities that collects and use this data.

5.1.2 European Union Privacy Laws for Location-based Services

The Europen Union has been more proactive in addressing the privacy issues related to LBS. There are three directives that address privacy in various aspects of a user's location data and LBS requests. These directives are:

Directive (95/46/EC) on processing of personal data. This directive is a general directive and applies to any personal data (including location data and LBS requests) that does not fall into the scope of any sector-specific directive.

Directive (2002/58/EC) on processing of personal data in e-communication. This directive, popularly known as e-Privacy Directive, applies to all the traffic data and location data that is processed in an electronic communication network. The traffic data means any data processed for the purpose of the conveyance of a communication on an electronic communication network.

Directive (2006/24/EC) on data retention. This directive regulate the storage and retention of traffic data in a communication network. It requires the service providers to store the traffic data (not the content of communication) for the purpose of investigation, detection and prosecution of serious crime. The required duration of storage is at least six months with a maximum of two years. It must be ensured that the retained data is provided only to the competent national authorities in specific cases and in accordance with national law.

All these directives together affects the privacy treatment of the location data and LBS requests in different types of LBS. The privacy of the location data is primarily regulated by the e-Privacy directive, while that for the LBS requests is regulated by the data retention directive and directive for privacy of personal data. These directives describe the following measures:

a. **Requirement of consent.** The LBS provider must obtain user's consent for using the location data to provide the location-based service. The consent means any freely given specific and informed indications by which the individual signifies its agreement to use its location data and LBS requests. Consent may be given for any individual use or by an overall consent for a variety of similar use.

b. **Right to withdrawal of consent.** If the users give their consent to the LBS provider to use their location data, and retain the LBS request data, they must have the ability to rescind the consent in the future. Its unclear though that once the user rescinds his consent, whether the LBS can use the previously collected data or not.

c. **Right of temporary refusal of processing.** In addition to revoking the consent for using their location data to provide location based services, the user must be able to temporarily suspend usage of their location data and the LBS requests.

The LBS provider must provide this mechanism free of charge, but the actual mechanism depends upon the service provider.

d. **Information to the subscriber or user.** The service provider must inform the users, prior to obtaining their consent, about (1) the type of data (location, LBS requests) that will be collected, (2) the purpose and the duration for which the data is collected and (3) if the data is going to be shared with a third party and how it is going to be used by the third party.

e. **Restricted data access.** Access to the location data and the LBS requests must be restricted to personnel acting under the authority of the communication service provider and the LBS provider. The data must be used strictly to provide the service and not for any other auxiliary purpose. For e.g. the service providers cannot even use the location data and LBS requests for the purpose of promoting its services.

f. **Sharing with third parties.** The LBS provider can share the location data (other than the one required for routing) with third parties provided they fully inform the users about the sharing and obtain their consent for doing so.

The directive leaves space for the member nations to interpret, adapt and implement them in accordance with their nation law. This is specially the case with countries that already have legal requirements on collection, use and sharing of location data and LBS requests.

Shortcomings

Even though the EU has clearly stated privacy requirements for the LBS provider, they still lag behind the state of the art data collection and usage practices. As an example, consider the requirement of anonymizing the personal data after it is not required for value added service. There are no specific requirements for the anonymization techniques or anonymized data. As a result, the LBS providers are free to choose the anonymization technique of their choice which may or may not preserve privacy of the user.

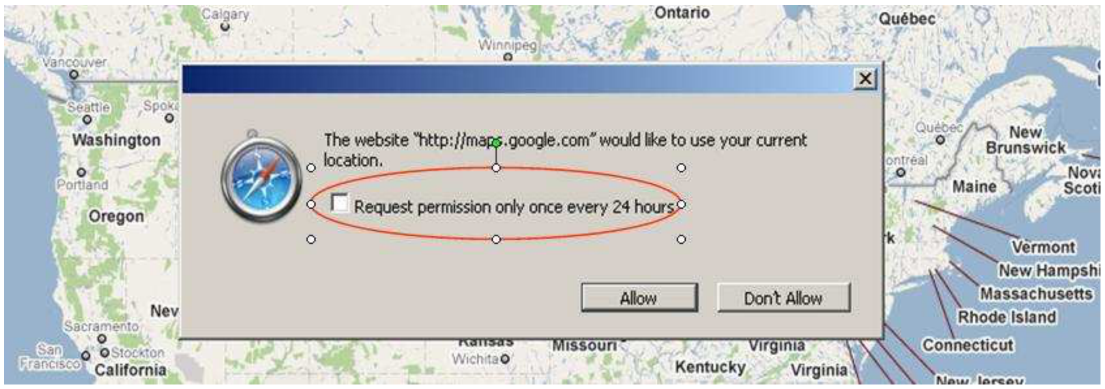
Even if the shortcomings of legal requirements are fixed, their jurisdiction is limited to the LBS providers that operate in that country or union. These legal requirements are cannot be enforced on LBS providers that operate from other countries or union that lack the legal framework for user privacy in LBS.

5.2 Privacy in commercial LBS Ecosystem

Most of the commercial LBS do just enough to comply with the legal requirements and/or recommendations of the country they operate from. Below we summarize the measures taken by commercial LBS to preserve user privacy in location-based services.

5.2.1 Obtaining user consent

Almost all the LBS providers deploy some mechanism to obtain before a user can use their services. This could be as a part of service agreement when users download LBS client to their mobile device or when they sign up for the service at the LBS provider's website or every time the user access the service. Figure 5.1 shows the pop-up in Safari (browser) that is used to obtain user consent when a location-enabled Web service wants to access their location.



The website "http://maps.google.com" would like to use your current location.

Request permission only once every 24 hours

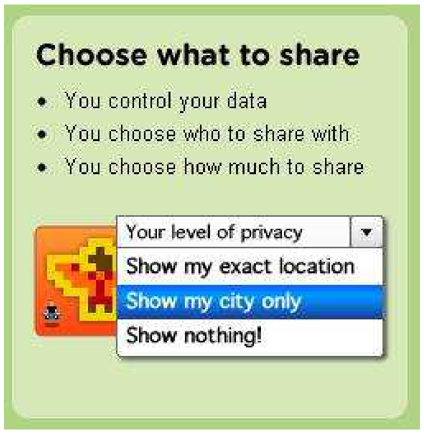
Figure 5.1: Obtaining user consent in Safari

5.2.2 Control over LBS data

Once a user sends an LBS request, the amount of control available to him over his LBS requests, varies across different LBS. While some applications (e.g. Google Maps) give *no control* to the users, some applications (e.g. FourSquare) allows the users to delete some or all the LBS requests sent in the past.

5.2.3 Anonymizers

As of now the only publicly available spatial cloaking anonymizer is Fire Eagle [fe] from Yahoo. It is primarily a location sharing platform that allows user to update their location in multiple subscription based LBS. The user sends their location updates to fire eagle that acts as the central repository. The user authorized LBS applications then can access his current location from the central repository. The anonymizer does not use k-anonymity as the privacy model but provides 3 levels of spatial cloaking (Figure 5.2): *exact location*, *city containing the location*, *nothing*.


Choose what to share

- You control your data
- You choose who to share with
- You choose how much to share

Figure 5.2: Fire Eagle Anonymizer

Chapter 6

Conclusion and Open Issues

6.1 Conclusion

We introduce the notion of policy-aware sender k-anonymity via spatial cloaking for LBS. The stronger privacy guarantee provides sender k-anonymity against an attacker who knows the complete details of the system used to provide the protection. We show that it is strictly stronger than policy-unaware sender k-anonymity. We show that problem of finding the optimum policy-aware sender k-anonymization of a set of users depends upon the type of cloak used. We show that it is NP-complete if the cloaks are circles whose centers are selected from a given set of points, but becomes PTIME if cloaks are picked among the quadrants of a quad-tree based partitioning of the map. We propose a polynomial time algorithm for optimum policy-aware k-anonymization of a set of mobile users and show that is practical and scales extremely well: it takes less than 1 second to anonymize 250k requests and up to 1 million requests in less than 4 seconds. We show that with policy-aware cloaking, the reduction in utility in comparison to a k-inside policy is reasonable: the average cloak area is at most 1.7 times the average area of the tightest cloaks used for policy-unaware anonymity. Thus this novel privacy guarantee strikes a pragmatic balance in the trade-off between strength of privacy guarantee, utility and running time of enforcement. We also show considerable amenability of the problem to parallelization which reduces the anonymization time while preserving optimal utility in virtually all cases. Using 16 servers, we can anonymize 1 million requests within 1 sec with only 1% divergence of the cost from the optimum.

But since the users have been using the LBS without any anonymization, some of their LBS requests that they consider private, have been logged by the LBS provider along with their other LBS requests. We describe the problem of offline trajectory-aware policy-aware sender k-anonymity for these LBS request logs. We show that trajectory-awareness requires stronger defense and extend the snapshot policy-aware sender k-anonymity to trajectory-aware attackers. We introduce bundles as a mechanism to publish the LBS requests logs with linkage information while preserving trajectory-aware and policy-aware sender k-anonymity. We show that trajectory-aware policy-aware anonymization is harder than snapshot policy-aware anonymization and that finding the optimum anonymization is *NP-Hard*, even with cloaks that are chosen among the quadrants of a quad-tree based partition of the map .This is significant since we showed earlier that for such cloak types optimum snapshot policy-aware k-anonymization is P-time. We show that problem of optimum trajectory-aware and policy-aware sender k-anonymity is approximable and propose a novel l-approximation algorithm to anonymize a LBS request log spanning user trajectories of length l. We describe novel optimizations to improve the average cost of anonymization of approximation algorithm. We implement our approximation algorithm and empirically show that it is practical and scales extremely well with the number of user trajectories: it takes less than 5 minutes to anonymize 1 million user trajectories of length 30 in the San Francisco Bay area.

6.2 Open Issues

The study of sender anonymity for LBS is in no way complete. We identified and addressed only some of the issues pertaining to sender anonymity, but there are still some open questions that needs to be addressed to move closer to the illusive goal of perfect anonymity. Next we describe some of these open problems. In addition our study have also raised some interesting questions in other related areas such as k-anonymization in data publishing.

6.2.1 Anonymity against background information

Policy-aware sender k-anonymity is an information theoretic guarantee, which means that the privacy guarantee depends upon the information with the user rather than his computation capabilities (like some of the cryptographic techniques). Therefore the solution proposed in this study may not provide sender k-anonymity against an attacker that has additional information about the user (e.g. some background information that allows it to use the query parameters to identify the sender). Thus there is a need of a framework to model the background information of a user that is available to an attacker and propose solution for sender k-anonymity against such attackers. The problem of optimum k-anonymity is known to be NP-hard, there is a need to study and identify cases or restriction when the problem is polynomial.

6.2.2 Privacy built into the LBS

Based on the study of various commercial LBS, their privacy and data retention policies and the privacy incidents associated with them (described in the Introduction), it is quite evident that user privacy in 1st generation of commercial LBS is an *after thought*, i.e. it is not inherent in the design or working of the LBS. In fact, the vested interests of entities in the LBS ecosystem, other than the sender, view the privacy preserving solutions as a deterrent to their services or businesses. For e.g. an LBS that provides free service to its users but integrates location-based ads in the response, will be wary of any change that would make the ads less relevant to the user. On top of it, there are financial considerations such as ”who should bear the responsibility and the cost of an anonymizer”. As a result, the commercial adaptation of the privacy preserving solutions has been negligible.

This is understandable for the 1st generation of LBS, given that there was neither a clear commercial viability for any particular LBS services delivery model nor a complete understanding of user privacy issues. But now that LBS has gained popularity and there is a prevalent model for LBS, the 2nd generation of LBS should be designed with inbuilt privacy features. For e.g. the LBS providers should have the capabilities to process LBS requests anonymized using spatial cloaking. As another example, the location server computes only k-anonymized location (a region) so that there is no requirement of an anonymizer. Such features would not only address privacy concerns of the LBS users but also reduces the number of trusted entities required for anonymous access to the LBS.

6.2.3 User education and feedback

In recent times the research community, consumer advocates and media have been considerably active in making the users aware of their location privacy, but the problem of sender anonymity has not received the same level of scrutiny. More efforts are required from these communities to educate the users about

- data collection practices of LBS providers i.e. what kind of data they collect, how long they keep it, whom they share it with.
- methods used by the LBS providers to obtain this data.
- potential misuse of the collected data.
- measures that could provide sender anonymity.

From the privacy research point of view, the last action item in the above list is very important because the success and widespread adoption of any privacy measure depends to some extent on the layman's understanding of that measure.

6.2.4 Policy-awareness in Data Anonymization

Our investigation of sender k-anonymity privacy guarantee in LBS has revealed that anonymity against a policy-aware attacker has been ignored or mishandled. Most solutions does not even consider a policy-aware attacker and thus fail to provide anonymity against them. Some consider such an attacker, yet as shown they fail to provide anonymity. Since the notion of k-anonymity has also been used for data anonymization, one must evaluate the data k-anonymization solutions to check if they preserve user privacy against a policy-aware attacker. While we are not aware of any study that considers a policy-aware attacker in data k-anonymity, there are certain variations of k-anonymity that consider attackers who are aware for certain properties of the k-anonymizing policy. In [WFWP07], the privacy guarantee defends against an attacker who knows that k-anonymizing policy in use tries to minimizes the data distortion.

Bibliography

[ABN08] Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 376–385, Washington, DC, USA, 2008. IEEE Computer Society.

[acl] ACLU of Northern California.
http://www.aclunc.org/issues/technology/location-based_services_time_for_a_privacy_check-in.shtml.

[AFK$^+$06] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. In *PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 153–162, New York, NY, USA, 2006. ACM.

[BKBL07] Ji-Won Byun, Ashish Kamra, Elisa Bertino, and Ninghui Li. Efficient -anonymization using clustering techniques. In Kotagiri Ramamohanarao, P. Radha Krishna, Mukesh K. Mohania, and Ekawit Nantajeewarawat, editors, *DASFAA*, volume 4443 of *Lecture Notes in Computer Science*, pages 188–200. Springer, 2007.

[BMWJ07] Claudio Bettini, Sergio Mascetti, X. Sean Wang, and Sushil Jajodia. Anonymity in location-based services: Towards a general framework. In *MDM '07: Proceedings of the 2007 International Conference on Mobile Data Management*, pages 69–76, Washington, DC, USA, 2007. IEEE Computer Society.

[Bow] Professor William Bowen. DIGITAL ATLAS OF CALIFORNIA.
http://130.166.124.2/CApage1.html.

[Bria] Thomas Brinkhoff. A framework for generating network-based moving objects.
http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator.

[Brib] Thomas Brinkhoff. A framework for generating network-based moving objects.
http://iapg.jade-hs.de/personen/brinkhoff/generator/.

[BS03] Alastair R. Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[BW10] Jeremiah Blocki and Ryan Williams. Resolving the complexity of some data privacy problems. 2010.

[BWJ05] Claudio Bettini, Xiaoyang Sean Wang, and Sushil Jajodia. Protecting privacy against location-based personal identification. In Willem Jonker and Milan Petkovic, editors, *Secure Data Management*, volume 3674 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2005.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24:84–90, February 1981.

[Cha88] D. Chaum. The dining cryptographers problem: unconditional sender and recipient untraceability. *J. Cryptol.*, 1:65–75, March 1988.

[CM07] Chi-Yin Chow and Mohamed F. Mokbel. Enabling private continuous queries for revealed user locations. In *10th International Symposium of Advances in Spatial and Temporal Databases*, volume 4605 of *Lecture Notes in Computer Science*. Springer, 2007.

[CT07] Chuang-Cheng Chiu and Chieh-Yuan Tsai. A k-anonymity clustering method for effective data privacy preservation. In *ADMA '07: Proceedings of the 3rd international conference on Advanced Data Mining and Applications*, pages 89–99, Berlin, Heidelberg, 2007. Springer-Verlag.

[cti] Best Practices and Guidelines for Location-Based Services.
http://files.ctia.org/pdf/CTIA_LBS_Best_Practices_Adopted_03_10.pdf.

[DHVZ09] Alin Deutsch, Richard Hull, Avinash Vyas, and Kevin Zhao. Policy-aware sender anonymity in location based services. Technical Report TR CS2009-0939, UCSD, 2009.

[DHVZ10] Alin Deutsch, Richard Hull, Avinash Vyas, and Kevin Keliang Zhao. Policy-aware sender anonymity in location based services. In Feifei Li, Mirella M. Moro, Shahram Ghandeharizadeh, Jayant R. Haritsa, Gerhard Weikum, Michael J. Carey, Fabio Casati, Edward Y. Chang, Ioana Manolescu, Sharad Mehrotra, Umeshwar Dayal, and Vassilis J. Tsotras, editors, *ICDE*, pages 133–144. IEEE, 2010.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[E91] Wireless 911 Services.
http://www.fcc.gov/cgb/consumerfacts/wireless911srvc.html.

[EGC$^+$10] 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 OSDI 2010*, October 2010.

[exa] Au's GPS cell phone shows how to get to McDonald's.
http://www.mobilemediajapan.com/headline2.asp?page=AU/KDDI.

[fac] Facebook Privacy Policy.
http://www.facebook.com/policy.php.

[fb] Facebook Places.
http://www.facebook.com/places/.

[fe] Fire Eagle: Take your location to the web.
http://fireeagle.yahoo.net/.

[fou] FourSquare Privacy Policy.
http://foursquare.com/legal/privacy.

[fs] Foursquare.
https://foursquare.com/.

[FWY05] Benjamin C. M. Fung, Ke Wang, and Philip S. Yu. Top-down specialization for information and privacy preservation. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 205–216, Washington, DC, USA, 2005. IEEE Computer Society.

[GDVM09] Aris Gkoulalas-Divanis, Vassilios S. Verykios, and Mohamed F. Mokbel. Identifying unsafe routes for network-based trajectory privacy. In *SDM*, pages 942–953. SIAM, 2009.

[GG03] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42, New York, NY, USA, 2003. ACM.

[GJ90] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[GKK$^+$08] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 121–132, New York, NY, USA, 2008. ACM.

[GL05] Bugra Gedik and Ling Liu. Location privacy in mobile systems: A personalized anonymization model. In *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pages 620–629, Washington, DC, USA, 2005. IEEE Computer Society.

[gla] Google Latitude.
http://www.google.com/mobile/latitude/.

[gls] Google Location Service.
http://www.google.com/intl/en/about/products/index.html.

[gma] Google Maps for Mobile.
http://www.google.com/mobile/maps/.

[gow] Gowalla Privacy Policy.
http://gowalla.com/privacy.

[gro] Groupon.
http://www.groupon.com.

[GRS99] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Commun. ACM*, 42:39–41, February 1999.

[hea] Chairman Franken's Opening Statement on Mobile Privacy.
http://www.youtube.com/watch?v=ZRHR7QK91fI.

[KGMP07] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Trans. on Knowl. and Data Eng.*, 19(12):1719–1733, 2007.

[lbs] Permanent Reference Document SE.23: Location Based Services.
http://www.gsmworld.com/documents/se23.pdf.

[LDR05] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 49–60, New York, NY, USA, 2005. ACM.

[LLV07] Ninghui Li, Tiancheng Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 106–115, April 2007.

[loc] Why Are Apple, Google, Tracking Your Phone.
http://abcnews.go.com/Technology/google-apple-track-users-location-information/story?id=13436330.

[loo] Loopt.
https://www.loopt.com/.

[LW08] Jun-Lin Lin and Meng-Cheng Wei. An efficient clustering method for k-anonymization. In *PAIS '08: Proceedings of the 2008 international workshop on Privacy and anonymity in information society*, pages 46–50, New York, NY, USA, 2008. ACM.

[MCA06] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new casper: query processing for location services without compromising privacy. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.

[MFD09] Noman Mohammed, Benjamin C.M. Fung, and Mourad Debbabi. Walking in the crowd: anonymizing trajectory data for pattern analysis. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 1441–1444, New York, NY, USA, 2009. ACM.

[MGKV06] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. ℓ -diversity: Privacy beyond κ -anonymity. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 24, Washington, DC, USA, 2006. IEEE Computer Society.

[MKA$^+$08] A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. pages 277–286, apr. 2008.

[MS84] N. Megiddo and K. J. Supowit. On the complexity of some common geometric location problems. *Siam J. Comput.*, 13(1):182–196, February 1984.

[mth] More Than.
http://www.morethan.com/.

[MW04] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–228, New York, NY, USA, 2004. ACM.

[NAS08] Mehmet Ercan Nergiz, Maurizio Atzori, and Yucel Saygin. Towards trajectory anonymization: a generalization-based approach. In *SPRINGL '08: Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, pages 52–61, New York, NY, USA, 2008. ACM.

[PBD09] Gianluca Della Vedova Paola Bonizzoni and Riccardo Dondi. The *k*-anonymity problem is hard. In *FCT*, pages 26–37, 2009.

[PW87] A Pfitzmann and M Waidner. Networks without user observability. *Comput. Secur.*, 6:158–166, May 1987.

[rob] Please Rob Me: Raising awareness about over-sharing.
http://pleaserobme.com/.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1:66–92, November 1998.

[Sal74] Jerome H. Saltzer. Protection and the control of information sharing in multics. *Commun. ACM*, 17(7):388–402, 1974.

[sho] Shopkick.
http://www.shopkick.com/.

[sky] Skyhook.
http://www.skyhookwireless.com/.

[spo] SpotRank by SkyHook.
http://www.skyhookwireless.com/spotrank/index.php.

[sta] Verizon Reports Sustained Revenue Growth and Continued Strong Cash Flows for 4Q and Full-Year 2008.
http://news.vzw.com/news/2009/01/pr2009-01-27.html.

[Swe02a] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, 2002.

[Swe02b] Latanya Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.

[tim] Numeric representation of dates and time.
http://www.iso.org.

[TM08] Manolis Terrovitis and Nikos Mamoulis. Privacy preservation in the publication of trajectories. In *MDM '08: Proceedings of the The Ninth International Conference on Mobile Data Management*, pages 65–72, Washington, DC, USA, 2008. IEEE Computer Society.

[WFWP07] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 543–554. VLDB Endowment, 2007.

[wsj] Your Apps Are Watching You.
http://online.wsj.com.

[XC07] Toby Xu and Ying Cai. Location anonymity in continuous location-based services. In *GIS '07: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*, pages 1–8, New York, NY, USA, 2007. ACM.

[YBLW09] Roman Yarovoy, Francesco Bonchi, Laks V. S. Lakshmanan, and Wendy Hui Wang. Anonymizing moving objects: how to hide a mob in a crowd? In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 72–83, New York, NY, USA, 2009. ACM.