MORGAN & CLAYPOOL PUBLISHERS

Privacy for Location-based Services

Gabriel Ghinita

SYNTHESIS LECTURES ON INFORMATION SECURITY, PRIVACY, AND TRUST

Elisa Bertino & Ravi Sandhu, *Series Editors*

Privacy for Location-based Services

Synthesis Lectures on Information Security, Privacy, & Trust

The Synthesis Lectures Series on Information Security, Privacy, and Trust publishes 50- to 100-page publications on topics pertaining to all aspects of the theory and practice of Information Security, Privacy, and Trust. The scope largely follows the purview of premier computer security research journals such as ACM Transactions on Information and System Security, IEEE Transactions on Dependable and Secure Computing and Journal of Cryptology, and premier research conferences, such as ACM CCS, ACM SACMAT, ACM AsiaCCS, ACM CODASPY, IEEE Security and Privacy, IEEE Computer Security Foundations, ACSAC, ESORICS, Crypto, EuroCrypt and AsiaCrypt. In addition to the research topics typically covered in such journals and conferences, the series also solicits lectures on legal, policy, social, business, and economic issues addressed to a technical audience of scientists and engineers. Lectures on significant industry developments by leading practitioners are also solicited.

Privacy for Location-based Services

Gabriel Ghinita

www.morganclaypool.com

ISBN: 9781627051491 paperback
ISBN: 9781627051507 ebook

DOI 10.2200/S00485ED1V01Y201303SPT004

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON INFORMATION SECURITY, PRIVACY, & TRUST

Lecture #4
Series Editors: Elisa Bertino, *Purdue University*
Ravi Sandhu, *University of Texas, San Antonio*
Series ISSN
Synthesis Lectures on Information Security, Privacy, & Trust
Print 1945-9742 Electronic 1945-9750

Privacy for Location-based Services

Gabriel Ghinita
University of Massachusetts, Boston

SYNTHESIS LECTURES ON INFORMATION SECURITY, PRIVACY, & TRUST
#4

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

Sharing of location data enables numerous exciting applications, such as location-based queries, location-based social recommendations, monitoring of traffic and air pollution levels, etc. Disclosing exact user locations raises serious privacy concerns, as locations may give away sensitive information about individuals' health status, alternative lifestyles, political and religious affiliations, etc. Preserving location privacy is an essential requirement towards the successful deployment of location-based applications. These lecture notes provide an overview of the state-of-the-art in location privacy protection. A diverse body of solutions is reviewed, including methods that use location generalization, cryptographic techniques or differential privacy. The most prominent results are discussed, and promising directions for future work are identified.

KEYWORDS

privacy, anonymity, security, homomorphic cryptography, PIR, differential privacy, spatial databases, GIS

Contents

Preface

The increasing popularity of mobile devices with Internet connectivity (e.g., 4G, Wi-Fi) and global positioning capabilities (e.g., GPS) have triggered the widespread development of many location-based applications. For instance, users are able to browse through maps of their nearby areas and to ask queries about points of interest in their proximity. The emergence of location-based social networks, or geosocial networks (GSN), allows users to share their whereabouts with friends, to find nearby contacts, and to provide/search for recommendations about points of interest that are close to their current geographical coordinates. Furthermore, users can act as mobile sensors in various location-centric crowd-sourcing scenarios, such as monitoring the levels of vehicle traffic congestion, or the levels of air pollution. However, such applications require users to disclose their locations in one form or another, which raises serious privacy concerns. With knowledge of user locations, a malicious adversary can stage a broad spectrum of attacks against individuals, from physical surveillance and stalking, to identity theft, to inferring sensitive information, such as the individual's health status, alternative lifestyles, political and religious affiliations, etc.

Preserving location privacy is an essential requirement towards the successful deployment of location-based applications. Currently, there are several related use-case scenarios in which some form of location sharing is necessary, and for which privacy must be preserved. First, users send location-based queries to un-trusted servers that store databases of points of interest. In this case, the privacy objective is to allow the users to retrieve nearby points of interest without having to disclose exact locations to the location service provider. Second, commercial entities that users trust with their locations, such as a telephone company, or an online GSN, wish to perform collaboratively a computation in the form of data matching with spatial predicates. Such computation is needed for data mining tasks that enhance the user experience by synthesizing profiles, or increase the companies' revenue. Each party will contribute its own dataset of locations, but only in anonymized form, and only the results of the computation should be revealed to each party, without gaining access to the raw dataset of its counterpart. Third, public entities such as a transportation company, or a road infrastructure monitoring network, collect large amounts of location data, either in the form of snapshot location samples, or in the form of trajectories, i.e., sequences of consecutive location readings corresponding to the same user. Such datasets, if released publicly, can be used for research or other public benefit purposes. However, the data needs to be sanitized, to prevent an adversary from associating location snapshots or trajectories with user identities.

These lecture notes provide an overview of the state-of-the-art in location privacy protection from the multiple perspectives outlined above. The notes review a diverse body of solutions, both in terms of design choices, as well as in terms of underlying technical approaches, ranging from anonymization techniques using location generalization, to cryptographic techniques, geo-

metric transformations, and finally the recently-emerging concept of differential privacy. The most prominent results are reviewed, their relative strengths and weaknesses are highlighted, and future directions of work that look most promising are identified.

Gabriel Ghinita
April 2013

Acknowledgments

The author was supported in part by National Science Foundation under grant CNS-1111512.

The author would like to thank Elisa Bertino for her invaluable mentorship during the collaboration on several location privacy research projects since 2008.

Some of the material presented in these lecture notes appeared originally in references [21, 22, 23, 26, 33]. Many thanks to Panos Kalnis for permission to use some of the materials in Chapter 3.

Gabriel Ghinita
April 2013

CHAPTER 1

Introduction

The increased popularity of mobile communication devices with embedded positioning capabilities (e.g., GPS) has generated unprecedented interest in the development of location-based applications. Users are able to browse through maps of their nearby areas and to ask queries about points of interest in their proximity. At the same time, users can generate their own content with geospatial tags; for instance many online photo sharing services allow geospatial annotations for images, and browsing photos based on their location is made possible.

Another area of application that has seen an impressive emergence recently is that of location-based social networks, or geosocial networks (GSN), that allow users to share their whereabouts with friends, to find nearby contacts, and to provide/search for recommendations about points of interest that are close to their current geographical coordinates. Probably the most prominent GSN to-date is Foursquare,[1] which already has 20 million subscribed users, and uses the concept of check-in. Using check-ins, users let their friends know their locations, and commercial rewards in the form of badges are given to frequent customers. When a user has the most number of check-ins for a particular venue over a given time period, the user gains a special distinction called mayorship, creating an exciting atmosphere among users that compete to reach a certain status, all made possible by sharing of locations. Other traditional social network providers have responded quickly with their own location-centric applications: Google launched the Google Latitude[2] app, whereas Facebook[3] acquired at the end of 2011 Gowalla, another early player in the realm of geosocial networks, and is now offering its own location-based app, Facebook Places.

The area of geospatial crowdsourcing has also gained in popularity in recent years. Mobile devices owners can act as mobile sensors in various location-centric crowd-sourcing scenarios, such as crisis management or environmental monitoring. In the case of natural disasters, users that are close to the crisis site can immediately gather location-dependent data and propagate it toward the authorities, long before the first responders even get to the site of the crisis. Crowd-sourcing is useful even in non-crisis scenarios, for instance to measure the levels of vehicle traffic congestion, the levels of air pollution, or to propagate instant information about the weather.

Another interesting class of applications is the study of trajectory traces. Consider a company that offers integrated payment services: for instance, the *Octopus* [1] payment system deployed in Hong Kong enables users to pay for transportation and day-to-day purchases with a single proximity card. As a result, a large amount of transaction logs which contain movement data are gathered. A

[1] https://foursquare.com/
[2] www.google.com/latitude
[3] www.facebook.com

third party company or government organization may wish to access the data and derive trajectory patterns useful to optimize traffic flow. The data owner is bounded by contractual obligations to guarantee user privacy. On the other hand, releasing the data can provide significant revenues. The challenge is to publish trajectories in a privacy-preserving fashion that still allows the derivation of meaningful results (e.g., finding which road segments are most frequently subject to traffic jams).

All the above exciting applications benefit from the availability and potential for sharing of location information. However, uncontrolled location sharing can have dire consequences, when location data falls in the hands of malicious entities. With knowledge of user locations, an adversary can stage a broad spectrum of attacks against individuals, from physical surveillance and stalking, to identity theft, to inferring sensitive information, such as the individual's health status, alternative lifestyles, political and religious affiliations, etc. Consider the following scenario: Bob uses his GPS enabled mobile phone to ask the query "Find the nearest hospital to my present location." This query can be answered by a *Location-Based Service* (LBS) in a public server (e.g., Google Maps), which maintains a database with *points of interest (POI)*. However, the LBS is *not* trusted. To preserve his privacy, Bob does not contact the LBS directly. Instead he submits his query via an intermediate trusted server which hides his ID (services for anonymous web surfing are commonly available nowadays). However, the query still contains the exact coordinates of Bob. One may reveal sensitive user data, such as religious affiliations or alternative lifestyles, by combining the location with other publicly available information (e.g., a telephone directory).

The early work of Gruteser and Liu [28] identifies three aspects of location information disclosure: position awareness, sporadic queries, and location tracking. Position awareness refers to the case where a device monitors an individual's location (e.g., an in-car GPS system), but no data is released to another party. The user's position is only used locally, to navigate a map for instance, hence no privacy threat occurs. The sporadic (or one-time) queries case refers to scenarios where a user reports his/her current location to a service provider, in order to find nearby points of interest (e.g., "find the closest restaurant"). Lastly, location tracking occurs in applications that require frequent updates of the user's position, e.g., traffic monitoring. Note that these disclosure scenarios do not always occur separately. For instance, both sporadic and frequent location updates may arise in the case of private LBS queries. If a user issues a continuous query, e.g., "report the location of the closest restaurant while I move," multiple locations (or a tentative trajectory) must be sent to the service provider. On the other hand, the duration of location reporting may be much shorter than in the case when an automobile acts as a mobile sensor and reports its coordinates and velocity readings for the entire duration when the ignition is turned on.

Another important aspect in location disclosing is related to the attacker capabilities. In [28], the authors discuss the concepts of *weak* and *strong* privacy. Weak privacy requires that no sensitive data should be *directly* disclosed to a party that is not trusted. In other words, if the current location of the user does not reveal any sensitive information, it is safe to disclose. This requirement may be sufficient if an attacker only gains access to sporadic location updates. On the other hand, if the attacker has access to a history of locations, additional information can be inferred. For instance,

if the trajectory of Bob includes along its way a hospital building, the attacker may associate him with a medical condition, even if Bob turns off his mobile device upon entering the hospital. In this case, *strong* privacy is required. Strong privacy disallows the publication of location snapshots which, although they do not represent a privacy violation by themselves, may be correlated to additional data to infer the presence of a user at a privacy-sensitive position. Anonymizing trajectory data is a representative example where strong privacy is necessary. Nevertheless, enforcing strong privacy must not have a significant negative impact on data accuracy, in the sense that the utility of the published data must be preserved.

These lecture notes provide an overview of the state-of-the-art in location privacy protection from multiple perspectives, by reviewing the requirements and characteristics of several different location sharing application scenarios. The area of location privacy solutions is a diverse one, both in terms of design choices, as well as in terms of the underlying technical approaches, which range from anonymization techniques using location generalization, to cryptographic techniques, geometric transformations, and finally the recently emerging concept of differential privacy. These notes review the most prominent results, highlight their relative strengths and weaknesses, and identify future directions of work that look most promising.

First, the problem of preserving query privacy in location-based services (LBS) is considered. Historically, this has been one of the first application settings where location privacy was studied, and is perhaps also the area where the most mature and robust solutions have been proposed. In this setting, users send location-based queries to un-trusted servers that store databases of points of interest. The privacy objective is to allow the users to retrieve nearby points of interest without having to disclose exact locations to the location service provider. Chapter 2 reviews techniques that employ spatial transformations for location protection, most often in the form of location generalization. While many such models have been proposed, and often the overhead of privacy can be reduced considerably and several scalable solutions are known, spatial transformations do not provide high levels of privacy, especially when adversaries have access to background knowledge. Chapter 3 introduces a more recent and provably secure category of protection methods for LBS queries which use cryptography. Private Information Retrieval (PIR) technology can be carefully combined with spatial encoding of datasets to allow content-based queries while providing privacy guarantees under the most adverse scenarios. However, such techniques tend to be expensive, and not practical for very large datasets. For this reason, Chapter 4 gives an overview of hybrid approaches that bring together the benefits of both spatial and cryptographic approaches, and can be used to answer both approximate and exact nearest-neighbor queries, the most frequent type of LBS queries.

Second, the problem of private matching of spatial datasets is reviewed. It is often the case that commercial entities that users trust with their locations, such as a telephone company, or an online GSN, wish to perform collaboratively a computation in the form of data matching with spatial predicates. Such computation usually takes the form of data mining tasks that enhance the user experience by synthesizing profiles, or increase the companies' revenue, and often rely on identifying locations in one of the datasets that are close to locations in the other dataset. Each party contributes

its own dataset of locations, but only in anonymized form, and only the results of the computation should be revealed to each party, without gaining access to the raw dataset of the counterpart. Chapter 5 outlines a solution that allows private matching of spatial datasets through a special type of geometric transformation which is robust against adversaries with background knowledge. The technique is also accurate, as it incurs only very few false positives, i.e., pairs of points that are found as a match even though the first point in the pair, corresponding to the first dataset, is not always within a threshold distance from the second point in the pair, corresponding to the second dataset. Interestingly, even in the case of false positives, the distance by which the method errs is small (typically a small constant fraction above the threshold distance).

Third, public entities such as a transportation company, or a road infrastructure monitoring network, collect large amounts of location data, typically in the form of trajectories, i.e., sequences of consecutive location readings corresponding to the same user. Such datasets, if released publicly, can be used for research or other public benefit purposes. However, the data needs to be sanitized, to prevent an adversary from associating trajectories to user identities. Chapter 6 surveys location generalization methods that sanitize trajectory data before publication, in order to prevent associations between users and sensitive locations.

Fourth, Chapter 7 reviews recent work that brings for the first time the novel and robust concept of *differential privacy* to the realm of location privacy. Differential privacy has the benefit of representing a *semantic* privacy model, robust and firmly grounded in statistical analysis, which has clear benefits over the ad-hoc, *syntactic* models that have been used for publication before it, such as location generalization. The main benefit of differential privacy is that it protects against adversaries with background knowledge. Although the work on differentially private location sharing is still at an early stage, the chapter reviews two techniques: the first one allows publication of individual location snapshots, in the form of indexed datasets that are built according to the principles of differential privacy, whereas the second shows how to publish information about location sequencing, which is required to publish databases of trajectories. While the accuracy of differential privacy still lacks that of the generalization counterparts, its superior privacy features, as well as encouraging results in boosting precision, make differential privacy the candidate of choice for the future of privacy-preserving location publication.

Finally, Chapter 8 concludes and identifies several open issues that represent interesting directions for future research.

CHAPTER 2

Privacy-Preserving Spatial Transformations

To preserve privacy, the exact location of users that send queries to Location-Based Services (LBS) must not be disclosed. Instead, location data is first perturbed, or encrypted. For instance, some existing techniques generate a few random fake locations and send a number of redundant queries to the LBS [36, 56] to prevent user identification. Other methods employ the concept of k-anonymity [48, 51], a well-established concept in the publication of microdata (e.g., hospital records). In the LBS domain, *spatial k-anonymity (SKA)* is enforced by generating a *Cloaking Region (CR)*—sometimes referred to as Anonymizing Spatial Region (ASR)—which includes the query source as well as $k - 1$ other users [20, 27, 33, 44]. Finally, some techniques obscure the location data using spatial [35] or cryptographic [23] transformations.

Achieving privacy incurs an additional overhead in processing queries: for instance, a larger number of queries need to be processed in the case of techniques that generate redundant requests. For spatial k-anonymity techniques, query processing is performed with respect to the CR, which is considerably more expensive than processing point queries. Therefore, a trade-off emerges between privacy and performance.

We can classify existing privacy-preserving spatial transformation techniques into two categories, according to the architecture they assume, namely: *(1)* two-tier spatial transformations, and *(2)* three-tier spatial transformations. Methods in the first category do not require any trusted third party, and the query anonymization is performed by the mobile user itself. Methods in the second category do assume the presence of a trusted third-party anonymizer server, and offer better protection against background knowledge attacks (e.g., an attacker may have additional information on user locations, from an external source). The trade-off is that methods in the second category generate more runtime overhead, because they require the users to costantly update their location with a central authority, and the algorithms used to generate protecting cloaking regions are more computationally expensive.

2.1 TWO-TIER SPATIAL TRANSFORMATIONS

Methods in this category involve only two parties at query time: the user and the LBS provider. Most of these methods[1] assume that no background knowledge is available to the attacker. A simple

[1] A notable exception is the *PROBE* [15] system which assumes that the attacker knows all sensitive locations.

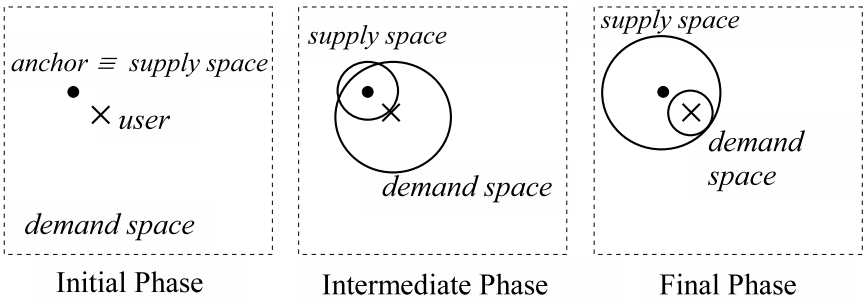


Figure 2.1: Incremental NN search process in SpaceTwist.

solution to query privacy is to generate a number of redundant queries for each real query. For instance, user u could generate r random "fake" locations, and send r redundant queries to the LBS, in addition to the actual query containing u's location. Such an approach is adopted in [36], where dummy locations are generated such that the resulting trajectories mimic realistic movement patterns. Dummy-generation algorithms can take into account movement parameters, such as velocity, and certain constraints, e.g., an underlying road network.

A more elaborate approach is *SpaceTwist* [56]: instead of generating a number of decoy locations beforehand, it performs a multiple-round, incremental nearest-neighbor query protocol, based on an *anchor* location. The anchor is initially set to a location randomly generated by the user. Throughout the query protocol, the user maintains two subsets of the dataspace: the *demand* and the *supply* spaces. The former consists of the space that needs to be covered by the issued queries, in order to ensure that the correct result is returned to the user, whereas the latter denotes the region of the space which is already covered. The client (the terms user and client are utilized interchangeably) knows both the demand space and the supply space, whereas the server knows only the supply space. Figure 2.1 gives an overview of query processing in SpaceTwist: initially, the demand space is set to the domain space, and the supply space contains only the anchor location. As points are retrieved from the server, the supply space expands. When a retrieved point is the closest point to the client seen so far, the results are updated, and the demand space shrinks. When the supply space eventually covers the demand space, it is termed final and the client is guaranteed to know its exact nearest neighbor.

The more recent work in [35] uses the Hilbert curve mapping [9] to transform the dataspace of points of interest. In a pre-processing (off-line) stage, a trusted entity transforms each POI p_i into its Hilbert value $H(p_i)$, and uploads the values to the LBS. The parameters of the transformations (e.g., curve orientation, scale, etc), are kept secret from the LBS, and represent the encryption key. To allow encoding of queries and decoding of results, users possess tamper-resistant devices that store the encryption key. At query time, the user u computes its transformed location $H(u)$ and

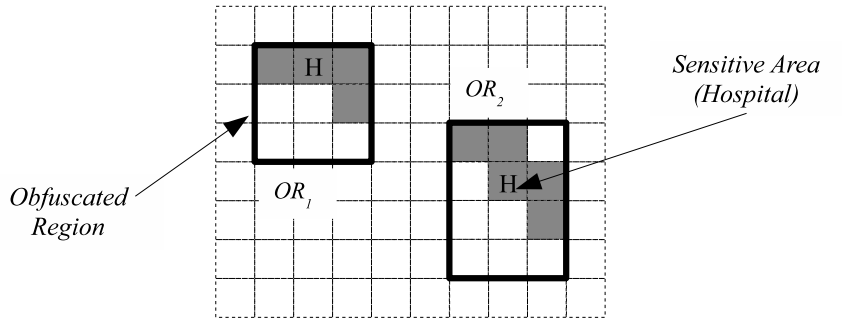


Figure 2.2: PROBE: association probability to "hospital" feature type is lower than 44%.

requests from the LBS the closest data value (in terms of 1D Hilbert values). Subsequently, the user decrypts the result by applying the inverse mapping H^{-1} to obtain the actual POI. The privacy of the solution relies on the large number of Hilbert curve parameter choices, and conjectures that it is computationally infeasible for the malicious attacker to decrypt Hilbert values to actual POI.

Nevertheless, the above solution is approximate in nature and does not provide any guarantee on the result accuracy.

The *PROBE* [15] system introduces a novel approach to location privacy, by preventing the association between users and sensitive locations (similar to the $\ell - diversity$ [40] concept from microdata anonymization). In PROBE, it is assumed that the attacker has access to all sensitive locations from a particular data space (e.g., a city, a country, etc). Sensitive locations are represented by *features*, which are classified into *feature types* (e.g., hospitals, restaurants, etc). In an off-line phase, an *obfuscated map* is constructed by partitioning the space into a set of disjoint regions such that the probability of associating each region with a certain feature type is bounded by a threshold. This process, called *obfuscation*, may require an additional trusted third party, but in the on-line phase (i.e., at query time) PROBE is a two-tier protocol. Figure 2.2 shows an obfuscated map with two obfuscated regions (OR_1 and OR_2): no region can be associated with the "hospital" feature type with probability higher than 44% (OC_1 contains nine grid cells in total, four of which are sensitive, and $4/9 = 0.44$).

Another interesting aspect about PROBE is that it can be extended to protect inference when users move across different obfuscated regions, as shown in [21]. Consider, for instance, the well-established division of U.S. territory into zip-code areas. The map is partitioned into disjoint regions, each of them covering an area of a few square miles. Or, at a finer granularity level, a city can be sub-divided into block regions. As the user moves, his/her location can be mapped to a city block identifier, and only the block identifier is disclosed. The *privacy requirement* in this case is not to allow an attacker to pinpoint the user location within a sub-region of a reported obfuscated region.

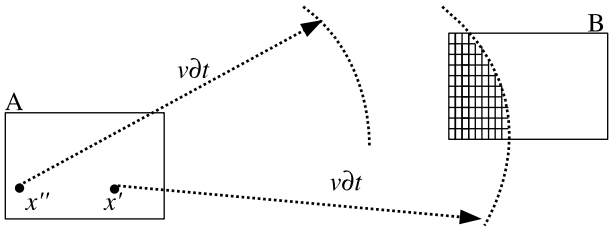


Figure 2.3: Attack model when user moves.

Figure 2.3 shows an example of two obfuscated regions A and B which are reported by user u at timestamps t_A and t_B, respectively. Without loss of generality, let $t_A < t_B$. Denote by v the maximum user velocity, and let $\delta t = |t_B - t_A|$.

The attacker may try to prune parts of A and B to pinpoint u in two ways:

(i) determine if there is any location $x \in A$ from which the user cannot reach some location $y \in B$, even by traveling at maximum speed v. Formally, an attack is successful iff.

$$\exists x \in A \; s.t. \; \forall y \in B, \; d(x, y) > v\delta t \tag{2.1}$$

In Figure 2.3, a user traveling from point x' is able to reach a point in the hatched region of B within time δt. However, if the initial location of u were x'', reaching B would not have been possible. Therefore, an attacker can rule out a subset of A as possible positions for u, hence privacy is breached

(ii) determine if there is any location $y \in B$ which the user cannot reach from some initial location $x \in A$, even by traveling at maximum speed v. Formally,

$$\exists y \in B \; s.t. \; \forall x \in A, \; d(x, y) > v\delta t \tag{2.2}$$

To prevent privacy breaches, it is necessary to ensure that none of Eq. (2.1) or (2.2) ever holds. This is equivalent to stating that the Hausdorff distance $d_{haus}(A, B) \leq v\delta t$. The Hausdorff distance is defined as follows:

$$d_{haus}(A, B) = \max\{h(A, B), h(B, A)\},$$

where

$$h(A, B) = \max_{p' \in A} \min_{p'' \in B} d(p', p'')$$

PROBE offers an amount of privacy which is superior to the other methods in this category. However, none of the two-tier spatial transformation solutions can prevent re-identification of the query source if an attacker has knowledge about specific user locations. For instance, if user u situated in a remote location issues a query (i.e., an outlier case), an attacker who knows that u is the only person residing in that area can associate u with the query, breaching user privacy. The next category of query anonymization methods deals with this issue.

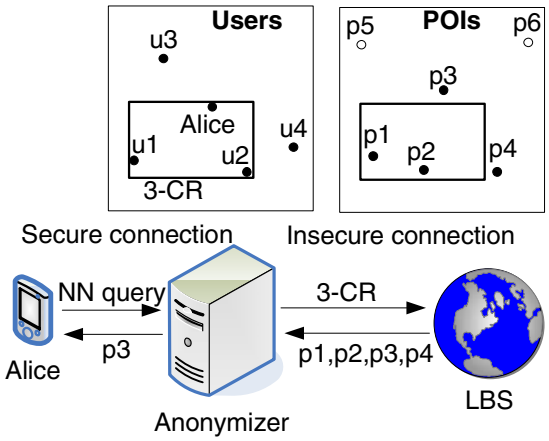


Figure 2.4: Spatial k-anonymity: three-tier architecture.

2.2 THREE-TIER SPATIAL TRANSFORMATIONS

Methods in this category implement the spatial k-anonymity paradigm: a cloaking region that contains $k-1$ users in addition to the query source (a k-CR) is generated, and the LBS processes the query with respect to the CR. Since all the k locations enclosed by the CR correspond to actual users (as opposed to "fake" locations in the previous category), the probability to identify the query source is at most $1/k$, even if the attacker has knowledge about exact user locations.

Most solutions in this category employ the three-tier architecture illustrated in Figure 2.4. A trusted centralized *anonymizer* acts as an intermediate tier between the users and the LBS. All users subscribe to the anonymizer and continuously report their location while they move. Each user sends his query to the anonymizer, which constructs the appropriate CR and contacts the LBS. The LBS computes the answer based on the CR, instead of the exact user location; thus, the response of the LBS is a superset of the answer. Finally, the anonymizer filters the result from the LBS and returns the exact answer to the user.

k-anonymity was first discussed in relational databases, where published data (e.g., census, medical) should not be linked to specific persons. Adam and Wortmann [3] survey methods for computing aggregate functions (e.g., *sum, count*) under the condition that the results do not reveal any specific record. Agrawal and Srikant [5] compute value distributions, suitable for data mining, in confidential fields. Recent work has focused on k-anonymity as defined in [48, 51]: a relation satisfies k-anonymity if every tuple is indistinguishable from at least k-1 other tuples with respect to a set of *quasi-identifier* attributes. Quasi-identifiers are attributes (e.g., date of birth, gender, zip code) that can be linked to publicly available data to identify individuals. Records with identical quasi-identifiers form an anonymized group. Two techniques are used to transform a relation to a k-anonymized one: *suppression*, where some of the attributes or tuples are removed, and *generalization*, which involves

replacing specific values (e.g., phone number) with more general ones (e.g., only area code). Both methods lead to information loss. Algorithms for anonymizing an entire relation, while preserving as much information as possible, are discussed in [6, 38]. Tao and Xiao [53] consider the case where each individual requires a different degree of anonymity, whereas Aggarwal [4] shows that anonymizing a high-dimensional relation leads to unacceptable loss of information due to the dimensionality curse. Finally, Machanavajjhala et al. [40] propose ℓ-diversity, an anonymization method that prevents sensitive attribute disclosure by providing diversity among the sensitive attribute values of each anonymized group.

In the domain of location-based services, cloaking was first introduced by Beresford and Stajano [7], who introduced the concept of a mix zone, which is similar to the k-CR, but do not provide concrete algorithms for spatial cloaking. Later on, Gruteser introduced *Interval Cloak* in [27], which is based on quadtrees. A quadtree [49] recursively partitions the space in quadrants until the points in each quadrant fit in a page/node. Figure 2.5 shows the space partitioning and a simple quadtree assuming that a node contains a single point. The anonymizer maintains a quadtree with the locations of all users. Once it receives a query from a user U, it traverses the quadtree (top-down) until it finds the quadrant that contains U and fewer than k-1 users. Then, it selects the parent of that quadrant as the k-CR and forwards it to LBS.

Assume that in Figure 2.5, U_1 issues a query with k=2. Quadrant[2] $\langle(0, 2), (1, 3)\rangle$ contains only U_1, and its parent $\langle(0, 2), (2, 4)\rangle$ becomes the 2-CR. Note that the CR may contain more users than necessary; in this example it includes U_1, U_2, U_3, although two users would suffice for the privacy requirements. A large CR burdens the query processing cost at the LBS and the network overhead for transferring a large number of candidate results from the LBS to the anonymizer. In order to overcome this problem, Gruteser and Grunwald [27] combine *temporal cloaking* with spatial cloaking, i.e., the query may wait until k (or more) objects fall in the user's quadrant. In the example, the query of U_1 will be executed when a second user enters $\langle(0, 2), (1, 3)\rangle$, in which case $\langle(0, 2), (1, 3)\rangle$ is the 2-CR sent to the LBS.

Similar to *Interval Cloak*, *Casper* [44] is based on quadtrees. The anonymizer uses a hash table on the user id pointing to the lowest-level quadrant where the user lies. Thus, each user is located directly, without having to access the quadtree top-down. Furthermore, the quadtree can be adaptive, i.e., contain the minimum number of levels that satisfies the privacy requirements. In Figure 2.5, for instance, the second level for quadrant $\langle(0, 2), (2, 4)\rangle$ is never used for $k \geq 2$ and can be omitted. The only difference in the cloaking algorithms of *Casper* and *Interval Cloak* is that *Casper* (before using the parent node as the k-CR) also considers the neighboring quadrants in the same level of the tree. Assume again that in Figure 2.5 U_1 issues a query and k=2. *Casper* checks the content of quadrants $\langle(1, 2), (2, 3)\rangle$ and $\langle(0, 3), (1, 4)\rangle$. Since the first one contains a user U_3, the 2-CR is set to $\langle(0, 2), (2, 3)\rangle$, which is half the size of the 2-CR computed by *Interval Cloak* (i.e., $\langle(0, 2), (2, 4)\rangle$).

In *Clique Cloak* [20], each query defines an axis-parallel rectangle whose centroid lies at the user location and whose extends are Δx, Δy. Figure 2.6 illustrates the rectangles of three queries

[2]Coordinates of the lower-left and upper-right points are used to denote a quadrant.

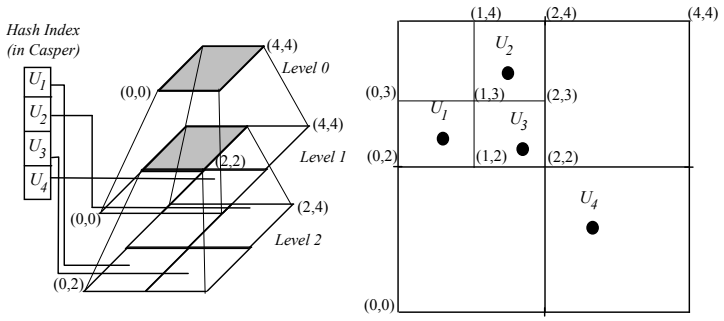


Figure 2.5: *Interval Cloak* and *Casper*.

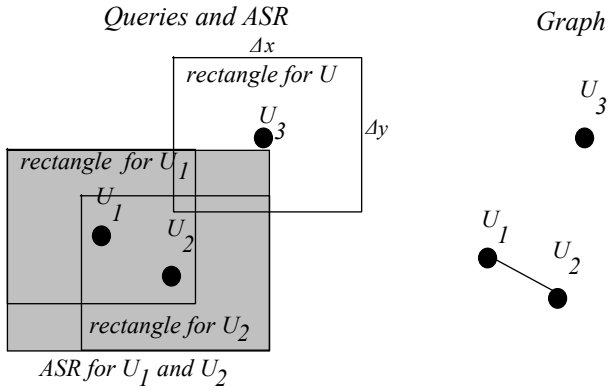


Figure 2.6: *Clique Cloak*.

located at U_1, U_2, U_3, assuming that they all have the same Δx and Δy. The anonymizer generates a graph where a vertex represents a query: two queries are connected if each falls in the rectangle of the other. Then, the graph is searched for cliques of k vertices and the minimum bounding rectangle (MBR) of the corresponding rectangles forms the ASR sent to LBS. Continuing the example of Figure 2.6, if k=2, U_1 and U_2 form a 2-clique and the MBR of their respective rectangles is forwarded so that both queries are processed together. On the other hand, U_3 cannot be processed immediately, but it has to wait until a new query (generating a 2-clique with U_3) arrives. *Clique Cloak* allows users to specify a temporal interval Δt such that, if a clique cannot be found within Δt, the query is rejected. The selection of appropriate values for Δx, Δy, Δt is not discussed in [20].

Probabilistic Cloaking [11] preserves the privacy of locations without applying spatial k-anonymity. Instead, the CR *(i)* is a closed region around the query point, which is independent of the number of users inside and *(ii)* the location of the query is uniformly distributed in the CR. Given a CR, the LBS returns the probability that each candidate result satisfies the query, based on

its location with respect to the CR. Finally, location anonymity has also been studied in the context of related problems. Kamat et al. [34] propose a model for sensor networks and examine the privacy characteristics of different sensor routing protocols. Hoh and Gruteser [30] describe techniques for hiding the trajectory of users in applications that continuously collect location samples. Chow et al. [13] study spatial cloaking in peer-to-peer systems.

Hilbert Cloak [33] uses the Hilbert space filling curve to map the 2-D space into 1-D values. These values are then indexed by an annotated B^+-tree. The algorithm partitions the 1-D sorted list into groups of k users (the last group may have up to $2k - 1$ users). For querying user u the algorithm finds the group to which u belongs, and returns the minimum bounding rectangle of the group as the CR. The same CR is returned for any user in a given group. Hilbert Cloak guarantees privacy for any distribution of user locations, but only for one-time (i.e., single-snapshot) queries.

Hilbert Cloak (*HC*) satisfies *reciprocity*, an important property that is *sufficient* for spatial k-anonymity.

Definition 2.1 Reciprocity Consider a user U issuing a query with anonymization degree k and associated k-CR A. A satisfies the reciprocity property iff *(i)* it contains U and at least k-1 additional users *(ii)* every user in A also generates the same CR A for the given k. The second condition implies that each user in A lies in the k-CRs of all other users in A.

In general, CRs generated by *Interval Cloak*, *Casper*, and *NNC* do not satisfy reciprocity as they violate condition *(ii)*. An optimal cloaking algorithm would partition the user population into CRs that have minimal sizes and obey the reciprocity property. However, calculating such an optimal partitioning is NP-Hard [42] and would require a fixed k by all queries. *HC* overcomes these problems by utilizing the Hilbert space-filling curve [43] to generate small (but not necessarily optimal) CRs for variable values of k. The Hilbert space filling curve transforms the 2-D coordinates of each user into an 1-D value $H(U)$. Figure 2.7 illustrates the Hilbert curves for a 2-D space using a 4×4 and 8×8 space partitioning. With high probability [45], if two points are in close proximity in the 2-D space, they will also be close in the 1-D transformation. A major benefit of Hilbert (and similar) curves is that they permit the indexing of multidimensional objects through one-dimensional structures (e.g., B-trees).

Given a query from user U with anonymity requirement k, *HC* sorts the Hilbert values and splits them into k-buckets. Each k-bucket has exactly k users, except for the last one which may contain up to $2{\cdot}k$-1 users. Let $H(U)$ be the Hilbert value of U and $rank_U$ be the position of $H(U)$ in the sorted sequence of all locations. *HC* identifies the k-bucket containing $rank_U$; that k-bucket is the k-CR. Figure 2.8 illustrates an example, where the user ids indicate their Hilbert order. For k=3, the users are grouped into three buckets (the last one contains four users). When any of U_1, U_2, or U_3 issues a query, *HC* will return the first bucket (shown shaded) as the 3-ASR.

HC is reciprocal because all users in the same bucket share the same k-CR; therefore, it guarantees spatial anonymity. Furthermore, it can deal with variable values of k by not physically storing the k-buckets. Instead, it maintains a balanced sorting tree, which indexes the Hilbert values.

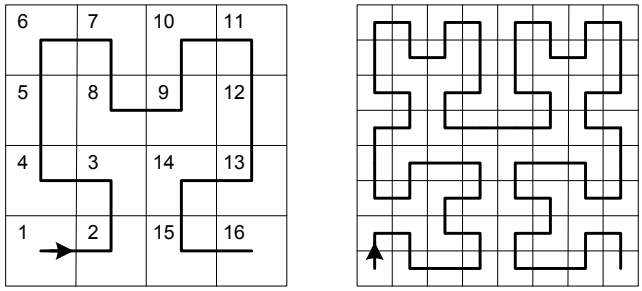


Figure 2.7: Hilbert curve.

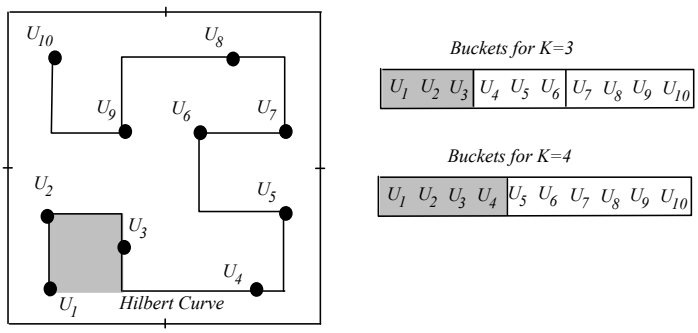


Figure 2.8: Example of *Hilbert Cloak*.

When a user U initiates a query with anonymization degree k, *HC* performs a search for $H(U)$ in the index and computes $rank_U$. From $rank_U$, the start and end positions defining the k-bucket that includes $H(U)$ are determined as follows:

$$start = rank_U - (rank_U \mod K), \qquad end = start + K - 1$$

The complexity of the in-order tree traversal is $O(N)$, where N is the number of indexed users. To compute $rank_U$ efficiently, an aggregate tree [52] is used, where each node w stores the number w_{count} of nodes in its left subtree (including itself). Using this data structure, $rank_U$ is computed in $O(logN)$ as follows: $rank_U$ is initialized to be equal to r_{count}, where r is the root, and a normal lookup for $H(U)$ is performed. For every node w visited, w_{count} is subtracted (resp., added) to $rank_U$ if a left (resp., right) branch is followed. The complexity of maintaining the aggregate information is $O(logN)$ because changes are propagated from the leaves to the root. Since the complexity of constructing the k-CR is $O(logN + K)$, whereas search, insert, and delete cost $O(logN)$, the data structure is scalable. Therefore, *HC* is applicable to a large number of mobile users who update their position frequently and have varying requirements for the degree of anonymity. Note that, while

the description assumes a main memory index, the technique can be easily extended to secondary memory by using B^+-trees.

2.3 DISCUSSION

Methods in the category of three-tier spatial transformations rely on the presence of other users to achieve spatial *k*-anonymity. These methods offer stronger privacy guarantees than two-tier techniques, with the exception of PROBE. The privacy features of PROBE and spatial *k*-anonymity methods are not directly comparable: PROBE does not achieve *k*-anonymity, but it does provide spatial diversity. On the other hand, three-tier techniques may not always prevent association of users to sensitive locations. For instance, it is possible for an entire CR to fall within a sensitive region (e.g., hospital). Therefore, the choice of paradigm (i.e., spatial anonymity vs. spatial diversity) ultimately depends on the specific application requirements.

CHAPTER 3

Cryptographic Approaches

A novel LBS privacy approach based on *Private Information Retrieval (PIR)* was introduced in [23]. Two such methods are proposed, which support approximate and exact private nearest-neighbor search, respectively. PIR protocols [12, 37] allow a client to privately retrieve information from a database, without the database server learning what particular information the client has requested. Most techniques are expressed in a theoretical setting, where the database is an n-bit binary string X (see Figure 3.1). The client wants to find the value of the i^{th} bit of X (i.e., X_i). To preserve privacy, the client sends an encrypted request $q(i)$ to the server. The server responds with a value $r(X, q(i))$, which allows the client to compute X_i.

The work in [23] builds upon the *computational* PIR ($cPIR$) protocol for binary data introduced in [37]. cPIR employs cryptographic techniques, and relies on the fact that it is computationally intractable for an attacker to find the value of i, given $q(i)$. Furthermore, the client can easily determine the value of X_i based on the server's response $r(X, q(i))$.

3.1 A PRIMER ON COMPUTATIONAL PIR

Computational PIR [37] relies on the *Quadratic Residuosity Assumption (QRA)*, which states that it is computationally hard to find the quadratic residues in modulo arithmetic of a large composite number $N = q_1 \cdot q_2$, where q_1, q_2 are large primes (see Table 3.1 for a summary of notations).

Define

$$\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N | gcd(N, x) = 1\}, \tag{3.1}$$

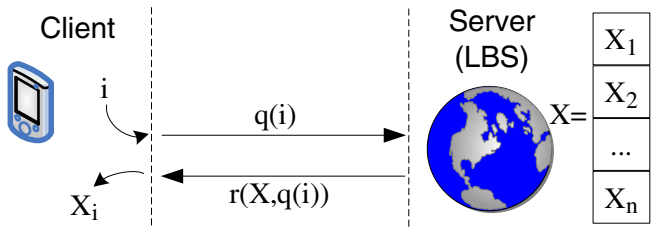


Figure 3.1: PIR Framework.

Table 3.1: Summary of notations

Symbol	Description
k	Modulus Bits
q_1, q_2	$k/2$-bit primes
$N = q_1 \cdot q_2$	Modulus
n	Number of Data Objects
m	Object Size (bits)
$t = \lceil \sqrt{n} \rceil$	PIR Matrix Dimension
$M_{1:t,1:t}[1:m]$	PIR Matrix (binary array)
$y_{1:t}$, array of k-bit numbers	PIR Query
$z_{1:t}[1:m]$, array of k-bit numbers	PIR Reply

the set of numbers in $\mathbb{Z}_N$ which are prime with N (gcd is the greatest common divisor). Then the set of *quadratic residues (QR)* modulo N is defined as:

$$QR = \{y \in \mathbb{Z}_N^* | \exists x \in \mathbb{Z}_N^* : y = x^2 \mod N\}. \tag{3.2}$$

The complement of QR with respect to $\mathbb{Z}_N^*$ constitutes the set of *quadratic non-residues (QNR)*.

Let

$$\mathbb{Z}_N^{+1} = \{y \in \mathbb{Z}_N^* | \left(\frac{y}{N}\right) = 1\}, \tag{3.3}$$

where $\left(\frac{y}{N}\right)$ denotes the Jacobi symbol [19]. Then, exactly half of the numbers in $\mathbb{Z}_N^{+1}$ are in QR, while the other half are in QNR. According to QRA, for $y \in \mathbb{Z}_N^{+1}$, it is computationally intractable to decide whether $y \in QR$ or $y \in QNR$. Formally, define the quadratic residuosity predicate Q_N such that:

$$Q_N(y) = 0 \Leftrightarrow y \in QR \tag{3.4}$$

Then, if q_1 and q_2 are $\frac{k}{2}$-bit primes, for every constant c and any function $C(y)$ computable in polynomial time, there exists k_0 such that

$$\forall k > k_0, \Pr_{y \in \mathbb{Z}_N^{+1}}[C(y) = Q_N(y)] < \frac{1}{2} + \frac{1}{k^c} \tag{3.5}$$

Hence, the probability of distinguishing between a QR and a QNR is negligible for large-enough k.

Let $t = \lceil \sqrt{n} \rceil$ and consider that the database X is organized as a square $t \times t$ matrix M (the matrix is padded with extra entries if n is not a perfect square). Let $M_{a,b}$ be the matrix element corresponding to X_i that is requested by the user u. u randomly generates modulus N (similar to a public key in asymmetric cryptography), and sends it to the server, together with query message $y = [y_1 \ldots y_t]$, such that $y_b \in QNR$, and $\forall j \neq b$, $y_j \in QR$.

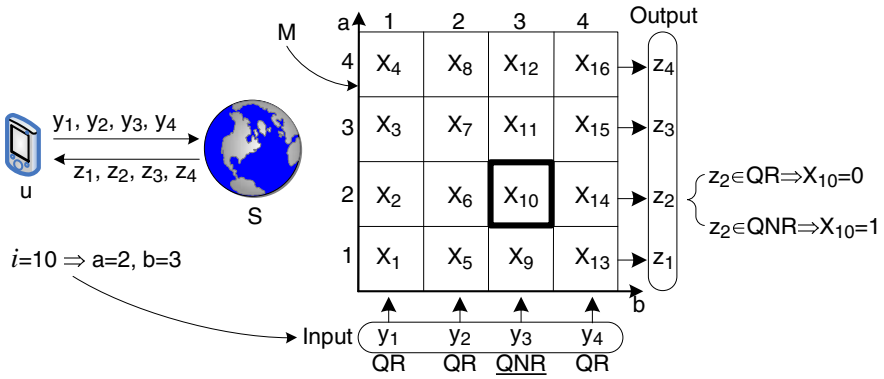


Figure 3.2: PIR example. u requests X_{10}.

The server computes for every row r of M the value

$$z_r = \prod_{j=1}^{t} w_{r,j} \tag{3.6}$$

where $w_{r,j} = y_j^2$ if $M_{r,j} = 0$, or y_j otherwise. The server returns $z = [z_1 \dots z_t]$. Based on the *Euler criterion*, u computes the following formula:

$$\left(z_a^{\frac{q_1-1}{2}} = 1 \mod q_1\right) \wedge \left(z_a^{\frac{q_2-1}{2}} = 1 \mod q_2\right) \tag{3.7}$$

If Equation 3.7 is *true*, then $z_a \in QR$ else $z_a \in QNR$. Since u knows the factorization of N, Equation 3.7 can be efficiently computed using the Legendre symbol [19]. The user determines the value of $M_{a,b}$ as follows: If $z_a \in QR$ then $M_{a,b} = 0$, else $M_{a,b} = 1$.

Figure 3.2 shows an example, where $n = 16$. u requests X_{10}, which corresponds to $M_{2,3}$. Therefore, u generates a message $y = [y_1, y_2, y_3, y_4]$, where $y_1, y_2, y_4 \in QR$ and $y_3 \in QNR$. The server replies with the message $z = [z_1, z_2, z_3, z_4]$. If $z_2 \in QR$ then u concludes that $X_{10} = 0$, else $X_{10} = 1$.

The protocol requires $O(n)$ multiplications at the server, and $O(\sqrt{n})$ communication cost. The latter can be reduced to $O(n^{\varepsilon}), 0 < \varepsilon < 1/2$, by applying the method recursively [37]. Although the recursive variation is asymptotically better than the basic one, our experiments revealed that the overhead of the recursion is not justified in practice.

The previous protocol retrieves privately one bit of information. The same idea can be extended to retrieve an object p_i which is represented as an m-bit binary string. Let D be a database containing n objects: $D = \{p_1, p_2, \dots p_n\}$. Again, the server generates a matrix M with the difference that each cell contains an m-bit object. Conceptually, this is equivalent to maintaining m matrices $M[1], M[2], \dots M[m]$, one for each bit of the objects. Assume that u requests object p_i.

Same as the 1-bit case, u sends a message $y = [y_1 \dots y_t]$. However, the server applies y to each one of the m matrices, resulting to m answer messages: $z[1], z[2], \dots z[m]$. u receives these messages and computes all m bits of p_i. The communication and computational cost increase to $O(m\sqrt{n})$ and $O(m \cdot n)$, respectively.

3.2 SPATIAL QUERIES WITH PIR

All existing location-privacy techniques studied in Chapter 2 rely on some form of cloaking, i.e., an exact location is perturbed to a spatial region. Nevertheless, although localization accuracy is reduced, a considerable amount of spatial information is still disclosed. This problem is more acute as cloaking algorithms strive for reduced-size CRs, to reduce processing overhead. For this reason, a number of attacks based on the geometrical properties of CR, or on the distribution of enclosed users, can be staged [33]. Furthermore, the CR can be used to narrow down the set of potential query sources; additional background information can then be used to pinpoint the query issuer.

PIR does not disclose *any* spatial information. As opposed to CR-based methods (which only perturb location, but still disclose the CR), no location information is disclosed. Instead, the data (i.e., POIs) are retrieved based on object index, by employing the provably private PIR protocol. This approach prevents any type of attack based on user location.

PIR also protects against correlation attacks. Assume that u asks a continuous query as he moves. Existing spatial tranformations generate one cloaking region CR_i per location, but all CR_i will include u. By intersecting the set of users in all CR_i, an attacker can identify u with high probability; this is called *correlation attack*. Note that this attack is possible because the CR reveals spatial information. Since the PIR framework does not reveal any spatial information, u is protected against correlation attacks.

PIR does not require any trusted third party, since privacy is achieved through cryptographic techniques. Spatial transformation techniques, on the other hand, need: (i) an anonymizer, which is a single point of attack, and (ii) a large set U' of subscribed users, all of whom must be trustworthy, since malicious users may collude to reveal the location of u. Furthermore, users in U' must accept the cost of sending frequent location updates to the anonymizer, even if they do not ask queries.

Finally, PIR reduces the number of disclosed POI. Existing techniques disclose a large set of candidate POIs. Since the database is a valuable asset of the LBS, users may be charged according to the result size. PIR techniques disclose far fewer POIs.

3.3 PROTOCOLS FOR APPROXIMATE AND EXACT NN WITH PIR

Note that, PIR protocols for binary data can support index-based queries, i.e., they retrieve the element with a given index i, whereas LBS queries are content-based, e.g., "find the closest POI to my location." The challenge in applying PIR to LBS privacy consists of finding effective methods to transform LBS queries into index-based queries.

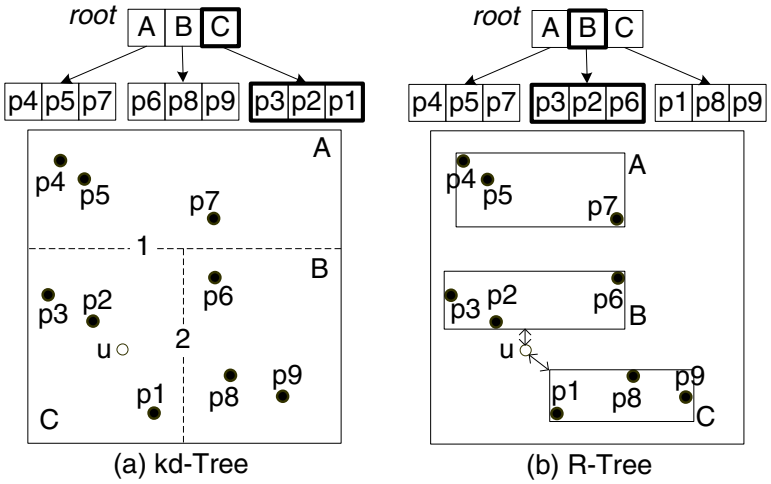


Figure 3.3: 2-D approximate NN.

First, an approximate NN protocol is outlined. The requirement of the protocol is to use a data structure that partitions the data into at most $\sqrt{n}$ buckets, each containing up to $\sqrt{n}$ POIs. Consider the case of the *kd*-tree [16]. The original insertion algorithm partitions the space either horizontally or vertically such that every partition contains one point. The modified algorithm to suit PIR is as follows: Let n' be the number of POIs in the current partition (initially $n' = n$), and let g be the number of remaining available partitions (initially, there are $\sqrt{n}$). Then, allow splits that create partitions e_1 and e_2 such that $|e_1| + |e_2| = n'$ and

$$\lceil |e_1|/\sqrt{n} \rceil + \lceil |e_2|/\sqrt{n} \rceil \leq g. \tag{3.8}$$

Subsequently, the algorithm is recursively applied to e_1 and e_2, with $\lceil |e_1|/\sqrt{n} \rceil$ and $\lceil |e_2|/\sqrt{n} \rceil$ remaining partitions, respectively. Out of the eligible splits, the most balanced one is chosen. In the example of Figure 3.3.a there are $n = 9$ POIs, and three available buckets. The points are split into regions A, which contains $|A| = 3$ POIs, and BC, which contains $|BC| = 6$ POIs. BC is further split into B (where $|B| = 3$) and C (where $|C| = 3$). The resulting *kd*-tree has two levels. The root contains regions A, B, C and the leaf level contains three nodes with three POIs each, which are arranged in a PIR matrix M. Since u is in region C, column 3 is retrieved; the NN is p_2.

As another case study, consider the R-tree. Originally, each node would store between $f/2$ and f objects, where f is the node capacity; internal nodes contain minimum bounding rectangles (MBR) which enclose the objects of their children. The R-tree construction algorithm is modified such that there are two levels and the root contains no more than $\sqrt{n}$ MBRs. Let n' be the number of POIs in the current partition. The original algorithm checks all possible partitionings with $|e_1| + |e_2| = n'$ POIs, along the x and y-axis. It selects the best one (e.g., lowest total area, or total perimeter, etc) and continues recursively. Figure 3.3.b shows an example where MBRs A, B, C contain three POIs

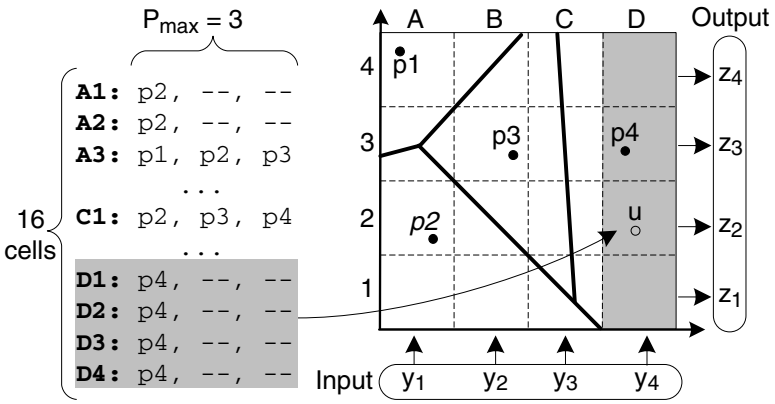


Figure 3.4: Exact nearest neighbor.

each. The leaf nodes are arranged in a PIR matrix M and u is closer to MBR B, therefore column 2 is retrieved and the NN is p_2.

Both 2-D methods return the approximate NN by retrieving $\sqrt{n}$ POIs. The communication cost is $O(\sqrt{n})$. Therefore, in terms of cost, they are the same. The only difference is the approximation error, which depends on the characteristics of the dataset (e.g., density, skew). The case studies of the *kd*-tree and R-tree demonstrate a general method for accommodating any partitioning in the PIR framework. The choice of the appropriate partitioning is dataset-specific.

Next, consider the case of exact NN queries, with the help of a method called *ExactNN* which returns the POI that is the exact nearest neighbor of user u. In a preprocessing phase, ExactNN computes the Voronoi tessellation [16] of the set of POIs (see Figure 3.4). Every Voronoi cell contains one POI. By definition, the NN of any point within a Voronoi cell is the POI enclosed in that cell. ExactNN superimposes a regular $G \times G$ grid on top of the Voronoi diagram. Then, for every cell c of the grid, it determines all Voronoi cells that intersect it, and adds the corresponding POIs to c. Hence, cell c contains all potential NNs of every location inside it. For example, Figure 3.4 depicts a 4×4 grid, where cell $A1$ contains $\{p_2\}$, cell $B2$ contains $\{p_2, p_3\}$, etc. During query processing, u learns the granularity of the grid; therefore he can calculate the cell that encloses his location (i.e., $D2$ in our example). Next, u issues a private request $PIR(D2)$; from the contents of $D2$ u finds his NN (i.e., p_4).

In contrast to ApproxNN methods, the objects of the PIR matrix M of ExactNN are not the POIs. Instead, each object in M corresponds to the contents of an entire grid cell c. For instance, our example contains 4 POIs (i.e., p_1, p_2, p_3, p_4), but M contains 16 objects, since there are 16 cells in the grid. In the previous section, n (i.e., the number of objects in M) was the same as the number of POIs. To avoid confusion, n still refers to the number of objects in M (i.e., $n = 16$ in the example) and we use $|POI|$ to denote the number of POIs.

All objects in M must have the same number of bits, otherwise the server may infer the requested cell based on the amount of bits transferred. Let P_{max} be the maximum number of POIs per grid cell. If a cell has fewer than P_{max} POIs, the server adds dummy POIs as placeholders. In our example, $P_{max} = 3$ because of cells $A3$ and $C1$. Therefore, all other cells are padded with dummy POIs. For instance, cell $A1$ becomes $\{p_2, -, -\}$. Recall from Table 3.1 that m denotes the number of bits of each object in M. Since there are P_{max} POIs in each grid cell, $m = |p_i| \cdot P_{max}$, where $|p_i|$ is the number of bits in the representation of each POI.

Since the number of objects in M is $n = G^2$, depending on the granularity of the grid, n may be larger or smaller than the number of POIs. P_{max} (hence m, too), also depends on G. Therefore, the communication and computational cost of ExactNN depends on G.

Exact NN Protocol

User u:	Initiate query
Server:	Send grid granularity G
User u:	Let b be the column that includes u
	$y = [y_1 : y_{\sqrt{n}}]$, $y_b \in QNR$, and $\forall j \neq b$, $y_j \in QR$
	Send y
Server:	Send $z[1:m] = [z_1 : z_{\sqrt{n}}][1:m]$
User u:	Let a be the row that includes u
	Discard dummy POIs in z_a
	Calculate distance to real POIs in z_a
	Return the exact NN

Figure 3.5: Protocol for exact NN.

The protocol for ExactNN is shown in Figure 3.5. It is similar to the ApproxNN protocol, with one difference: Let $\langle a, b \rangle$ be the cell that contains u, where a is the row and b the column. u issues a private request $PIR(\langle a, b \rangle)$. Recall that, in addition to $\langle a, b \rangle$, the byproducts of this request are the POIs of the entire column b. ApproxNN would utilize the extra POIs to improve the approximation of the result. On the other hand, the extra results are useless for ExactNN, since the exact NN is always in $\langle a, b \rangle$. A possible concern is that ExactNN reveals to the user $G \cdot P_{max}$ POIs, which may be more than those revealed by ApproxNN. In practice, however, this is not a problem because column b includes many duplicates. For example, cells $D1$, $D2$, $D3$, $D4$ in Figure 3.4 all contain the same POI p_4; therefore the request $PIR(D2)$ reveals only p_4 to the user.

3.4 COMPARISON WITH GEOMETRIC TRANSFORMATIONS

Supporting private LBS queries is achieved with an additional overhead in terms of computational and communication cost. Geometric and cryptographic methods provide various trade-offs between privacy and performance. Both aspects of this trade-off are discussed next.

The Privacy Aspect

Two-tier spatial transformations provide the least amount of privacy. For instance, dummy-generation offers no protection against attackers that possess background knowledge about user locations. Furthermore, exact locations are disclosed to the LBS, which is undesirable, since an attacker can learn that one of these locations corresponds to the actual user. SpaceTwist achieves slightly better privacy protection, because it does not disclose exact user locations. Still, an attacker that has knowledge on the user distribution within the supply space could infer the identity of the query source.

PROBE offers the strongest privacy features among the two-tier spatial methods. It assumes that the attacker has knowledge on all sensitive feature locations, and prevents the user-sensitive location association. In the worst case, however, an attacker may be able to associate a user with a sensitive query, since an obfuscated region may contain no other users in addition to the query source.

Methods in the category of three-tier spatial transformations do prevent user re-identification, because cloaked regions contain at least k actual users. Still, a cloaking region can have reduced extent (in the worst case, it can degenerate to a point). Therefore, an attacker may learn the whereabouts of the query source (and a number of $k - 1$ other users), and associate the locations of all users in the CR with a sensitive feature, compromising their privacy. No direct comparison can be made between PROBE, which offers spatial diversity, and three-tier methods that provide spatial anonymity. The choice of one method over another depends on the privacy requirements of each specific scenario. Furthermore, none of the geometric transformations methods is able to protect user privacy for continuous queries (i.e., over multiple snapshots of locations).

The PIR-based method in [23] provides full-featured privacy, under all attack models, since no information about user location is disclosed. Therefore, no association between users and sensitive locations can be performed. Even in the case when the attacker knows the exact locations of all users (from an external source), the association between users and queries is still prevented. The privacy guarantees hold for continuous queries (i.e., moving users) as well.

The Performance Aspect

The main overhead incurred by the dummy-generation scheme consists of processing redundant queries. However, there is only one round of processing. SpaceTwist also requires the processing of several point queries, but the process consists of multiple user-server interaction rounds, which can lead to increased response time.

PROBE and spatial anonymity methods incur a similar overhead in terms of query processing: a region query is processed in each case (corresponding to either a cloaked or an obfuscated region). Depending on the particular application scenario (e.g., density of sensitive locations and density of users), the relative performance of the two types of methods may vary. However, PROBE has the additional advantage that at query time, there is no overhead associated to obfuscated region generation, which is done off-line. In contrast, spatial anonymity methods require on-line cloaked region generation and frequent updates of user locations.

Finally, the PIR cryptographic-based approach may incur significant processing overhead, linear to the number of POI. As shown in [23], performance can be improved through a number of optimizations, such as re-using partial computation results, and parallelization. Still, the overhead is likely to exceed that of spatial transformation methods. Figure 3.6 shows a graphical representation of the discussed methods with respect to the privacy-performance trade-off achieved.

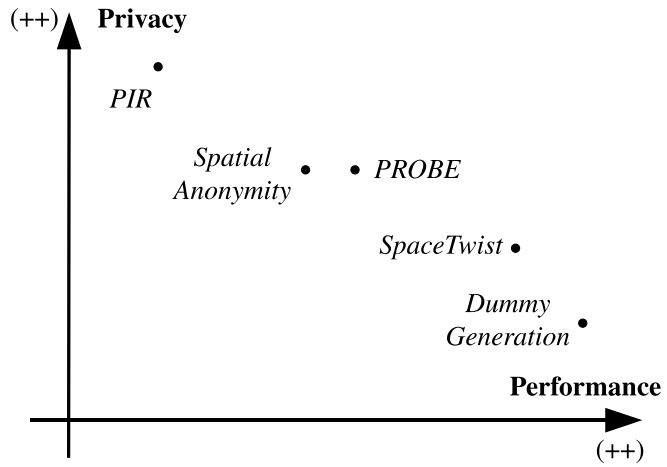


Figure 3.6: Privacy-Performance Tradeoff in LBS Privacy.

CHAPTER 4

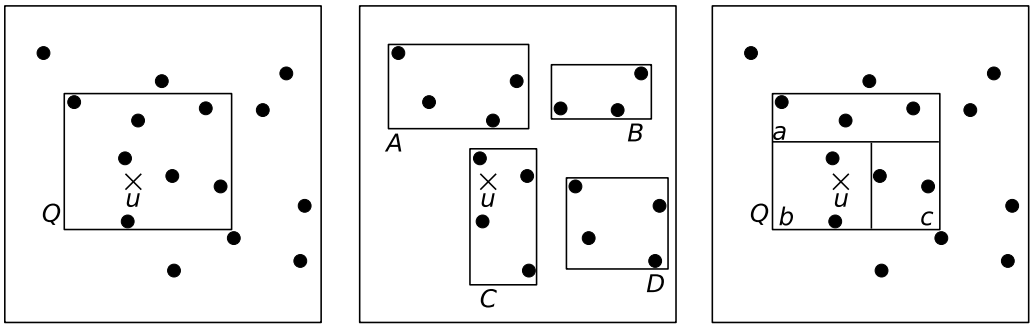
Hybrid Approaches

So far, we looked at geometric and cryptographic transformations in isolation, and we evaluated their relative trade-offs in terms of two important criteria: *privacy* and *performance*. With respect to privacy, the cryptographic approach (PIR) offers strong guarantees for both one-time, as well as repetitive (i.e., continuous) queries. Furthermore, PIR does not require trusted components, such as anonymizer services or other trusted users. On the other hand, CR methods operate under a more restrictive set of trust assumptions, but are considerably more efficient in terms of computational and communication overhead. The cryptographic elements incorporated in PIR require powerful computational resources (e.g., parallel machines), and high-bandwidth communication channels.

However, there is a third, equally important dimension in evaluating techniques for private location queries: the amount of protection provided to the database. In practice, it is important to control tightly the amount of POI disclosure, since the POI dataset represents a valuable asset to the service provider. For instance, consider that Bob asks the query: "find the nearest restaurant to my current location." The location server (LS) may reward Bob with certain discounts, in the form of electronic coupons (e.g., digital gift card codes) that are associated with each POI. If the user is billed on a "per-retrieved-POI" basis, then a large number of results will increase the cost of using the service. On the other hand, if the LS offers the service with no charge to the user (e.g., advertisement-generated income), then users could abuse the system by redeeming a large number of coupons. This causes the LS to lose its competitive edge, and to cease providing the service.

To illustrate the limitations of pure geometric and cryptographic approaches, consider the example of Figure 4.1, where the location server stores a database D of 15 POI (marked as full dots). User u asks a query for the nearest POI. If location cloaking is used (Figure 4.1(a)), the user will retrieve all the seven POI enclosed by query CR Q. As CRs grow large, location cloaking methods may disclose a large fraction of the database (possibly linear to $|D|$). On the other hand, the NN protocol from [23] does not use CRs. Instead, the dataset is partitioned into rectangular tiles $A \ldots D$, containing at most $\lceil\sqrt{15}\rceil = 4$ POI each (Figure 4.1(b)). The boundaries of the tiles are sent in plain text to u, who determines that his/her location is enclosed by tile C. Only the POIs in tile C are revealed to u through a PIR request. This method discloses $O(\sqrt{|D|})$ exact POI locations. However, revealing the tile boundaries may result in additional disclosure of POI locations, especially if the tiles have small spatial extent.

One can gain many advantages by employing *hybrid approaches* [22], consisting of a combination of geometric and cryptographic transformations. Hybrid techniques can be used for both approximate and exact NN queries. Figure 4.1(c) provides a brief illustration of how the hybrid two-step method works for approximate queried. The CR Q is sent to the LS, which determines a set of



(a) Location Cloaking (b) PIR (c) Hybrid

Figure 4.1: Benefit of the hybrid approach.

fine-grained tiles $\{a, b, c\}$ that cover the query area. A constraint is imposed that each tile encloses at most a constant number F of POI (a system parameter). The boundaries of the tiles are not sent to the user. Instead, the user and the LS engage in a cryptographic protocol that privately determines which one of the tiles encloses the location of u. At the end of the protocol, the LS learns nothing about the user location (except that u is inside Q), whereas the user only learns the identifier of the tile that encloses u (but not the boundaries of any of the tiles). Finally, the user requests through PIR the contents of the enclosing tile (in this case, b). The hybrid approach has two benefits: first, it controls strictly the amount of POI disclosed, which is bounded by a constant. This improvement is clearly superior to location cloaking and pure-PIR approaches, which disclose $O(|D|)$ and $O(\sqrt{|D|})$ POI, respectively. Second, the hybrid approach incurs considerably less overhead than the pure PIR method, since the cryptographic protocol is applied only on a partition of the database.

4.1 PRIVACY MODEL

The hybrid approach can be used in [22] conjunction with any of the location cloaking methods from [15, 20, 27, 28, 33, 44]. For instance, CRs can be built according to the spatial k-anonymity paradigm [20, 27, 33, 44], which requires that at least k distinct user locations must be enclosed by the CR. Alternatively, CRs can be determined based on user-specified sensitivity thresholds with respect to a set of sensitive feature types [15, 28]. Given a CR as an input to the hybrid technique, the focus is on two aspects: *(i)* how to efficiently perform PIR with respect to dynamically generated CRs, and *(ii)* how to control tightly the amount of disclosed POIs.

4.2 SYSTEM OVERVIEW

The system architecture is shown in Figure 4.2. The system model is flexible, and can accommodate several distinct solutions for creating input CRs. For instance, users can cloak their locations

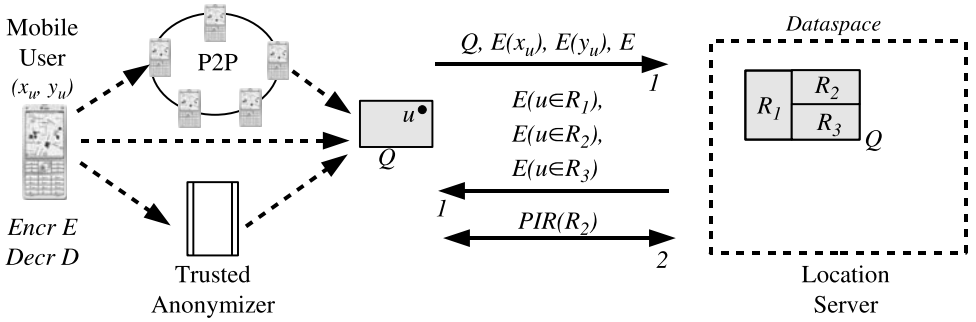


Figure 4.2: System architecture.

by themselves, as considered in [15, 28]. Alternatively, users can send their queries to a trusted anonymizer service which creates the CRs [20, 27, 33, 44]. Or, users can build CRs in a collaborative fashion [13, 24, 25].

Given the query CR Q, the LS returns the NN POI of the user by executing a two-round protocol, as shown in Figure 4.2. In the first round (arrows labeled 1), the user generates an encryption (E)/decryption (D) key pair, which are part of a homomorphic encryption family, such as Paillier [46]. The user sends to the LS the query CR Q, together with the encryption (i.e., public) key E and the encrypted user coordinates $E(x_u)$ and $E(y_u)$.

The LS processes the range query with argument Q and partitions the result set into a set of disjoint regions. In the case of approximate NN queries, these regions are rectangular tiles. In the case of exact NN queries, the regions are convex polygons representing all Voronoi cells in the POI dataset tessellation that intersect query Q. For brevity, the example in Figure 4.2 only shows the rectangular tiles for the approximate queries, but the concept used for exact queries is similar. Each rectangular region contains a number of POI bounded by constant F, which is a system parameter. For the given query, the set of tiles $\{R_1, R_2, R_3\}$ is obtained. The LS evaluates privately, using the properties of Paillier homomorphic encryption, the enclosure condition between point (x_u, y_u) and the resulting tiles. The encrypted evaluation outcome is returned to the user, who will decrypt and find which of the given rectangles encloses its location, in this case R_2. The private point-rectangle enclosure evaluation is necessary because the resulting query result tiles can be arbitrarily small. Sending these tiles in plain text to the user (as it is done in [23], with the root of the two-level index) would give away excessive information about the distribution of POI. Finally, in the second round of the protocol, the user issues a private request for the contents of R_2, and determines which of the retrieved POI is closest to his/her location.

At the foundation of the hybrid solutions for approximate and exact NN queries are two cryptographic protocols for private evaluation of point-in-rectangle enclosure, and point-in-convex-polygon enclosure, respectively. Both protocols rely on Paillier public-key homomorphic encryption scheme introduced in [46]. Paillier encryption operates in the message space of integers $\mathbb{Z}_N$, where

N is a large composite modulus. Denote by D and E the decryption and encryption functions, respectively. Given the ciphertexts $E(m_1)$ and $E(m_2)$ of plaintexts m_1 and m_2, the ciphertext of the sum $m_1 + m_2$ can be obtained by multiplying individual ciphertexts:

$$D(E(m_1) \cdot E(m_2)) = (m_1 + m_2) \mod N \tag{4.1}$$

In addition, given ciphertext $E(m)$ and plaintext $r \in \mathbb{Z}_N$, we can obtain the ciphertext of the product $r \cdot m$ by exponentiation with r, as follows:

$$D(E(m)^r) = r \cdot m \mod N \tag{4.2}$$

Furthermore, Paillier encryption provides semantic security, meaning that encrypting the same plaintext with the same public key E twice will result in distinct ciphertexts. Therefore, the scheme is secure against chosen plaintext attacks.

4.3 PRIVATE EVALUATION OF POINT-RECTANGLE ENCLOSURE

The two-party protocol is run between parties A and B and determines privately whether a given point p owned by A is enclosed in a rectangle R owned by B. The protocol protects the privacy of both parties involved. Specifically, A learns only if the point p is enclosed by R, but does not find any additional information about R. In addition, B does not learn any information about the point p of A. This can be achieved by privately evaluating the difference between the user coordinates and the boundary coordinates of rectangle R. Furthermore, to prevent leakage of POI locations, only the sign of the difference should be revealed to the user, and not the absolute value.

Assume that parties A and B hold two numbers a and b, respectively. Paillier encryption allows the computation of the ciphertext of sums based on the ciphertexts of individual terms. However, only the addition operation is supported, and not subtraction. Furthermore, the message space $\mathbb{Z}_N$ consists of positive integers only, hence the trivial solution of setting $m_1 = (-a)$, $m_2 = b$ and computing $E(m_1) \cdot E(m_2) = E(b - a)$ is not suitable. For this purpose, the message space is interpreted as the complement arithmetic representation for N-bit integers.

Assume that $a, b \in \mathbb{Z}_{N'}$, where $N' < N$. Party A computes $m_1 = N - a$ and sends $E(m_1)$ to B, who in turn sets $m_2 = b$, and determines

$$E(m_3) = E(m_1) \cdot E(m_2) = E(m_1 + m_2) = E(N + (b - a)) \tag{4.3}$$

Party B returns $E(m_3)$ to A who decrypts the message and learns the value of $m_3 = N + (b - a)$. The difference $b - a$ can be computed from m_3 as shown in Figure 4.3.

Let $I_1 = \{0, 1, \ldots, N'\}$ and $I_2 = \{N - N', \ldots, N - 1\}$. If $(b - a) \geq 0$, then $m_3 \in I_1$, otherwise $m_3 \in I_2$. To correctly interpret the result, it is necessary that $I_1 \cap I_2 = \emptyset$. A sufficient condition to ensure that the two intervals are disjoint is

$$[(N' - 0 + 1)] + [(N - 1) - (N - N') + 1] \leq N \Leftrightarrow N' \leq \left\lfloor \frac{N - 1}{2} \right\rfloor \tag{4.4}$$

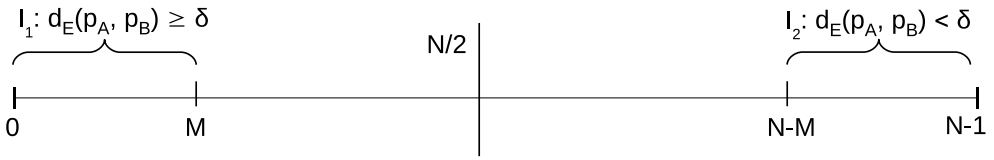


Figure 4.3: Determining the value of $b - a$.

Party A determines that

$$b - a = \begin{cases} m_3, & 0 \leq m_3 \leq N' \\ -(N - m_3), & N - N' \leq m_3 \leq N - 1 \end{cases} \tag{4.5}$$

The protocol requires only one round of communication. Note that A can immediately learn from $(b - a)$ the value of b.

The protocol for evaluating $(b - a)$ is further modified to disclose only $sign(b - a)$, without revealing any additional information about b. The main idea is to multiply m_3 in the previous protocol by a random blinding factor, such that the absolute value of $(b - a)$ can no longer be reconstructed by A. Consider random integer ρ uniformly distributed in the set $\{1, 2, \cdots, M\}$, such that

$$M \leq \left\lfloor \frac{N-1}{2N'} \right\rfloor \tag{4.6}$$

Instead of sending $E(m_3)$ back to A, B sends $E(m_4)$ obtained through exponentiation with plaintext ρ:

$$E(m_4) = E(m_3)^{\rho} = E(\rho \cdot m_3) = E(\rho \cdot (N + b - a)) \tag{4.7}$$

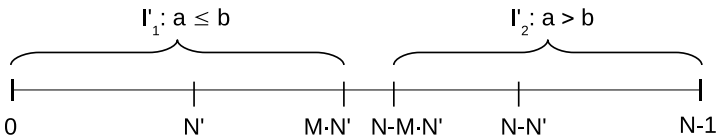


Figure 4.4: Private evaluation of $sign(b - a)$.

The value of $sign(b - a)$ can be computed from m_4 as shown in Figure 4.4. In a similar manner to the protocol for difference, let $I'_1 = \{0, 1, \ldots, M \cdot N'\}$ and $I'_2 = \{N - M \cdot N', \ldots, N - 1\}$. If $(b - a) \geq 0$, then $m_4 \in I'_1$, otherwise $m_4 \in I'_2$. This time, the condition $I'_1 \cap I'_2 = \emptyset$ is equivalent to

$$[(M \cdot N' - 0 + 1)] + [(N - 1) - (N - M \cdot N') + 1] \leq N \Leftrightarrow N' \leq \left\lfloor \frac{N-1}{2M} \right\rfloor \tag{4.8}$$

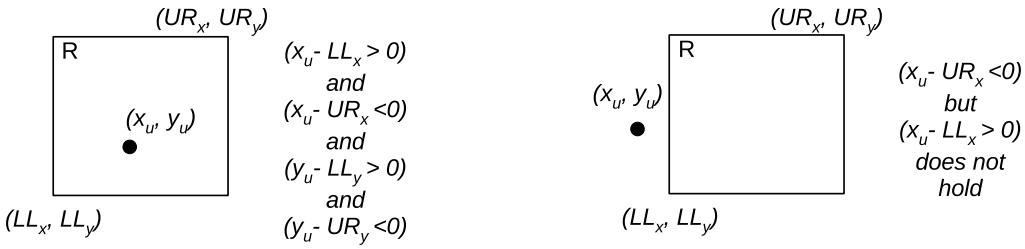


Figure 4.5: Arithmetic conditions to determine point-rectangle enclosure.

hence the requirement in Eq. (4.6). Party A determines that

$$sign(b-a) = \begin{cases} +1, & 0 & \leq m_4 \leq & M \cdot N' \\ -1, & N - M \cdot N' & \leq m_4 \leq & N-1 \end{cases} \tag{4.9}$$

The proof of Eq. (4.9) is immediate: if $(a \leq b)$, then $0 \leq m_3 \leq N'$, and therefore $0 \leq \rho \cdot m_3 \leq M \cdot N'$. On the other hand, if $(a > b)$ we have $N - N' \leq m_3 < N$, therefore

$$M(N-N') \mod N \leq M \cdot m_3 < N \Leftrightarrow (N - M \cdot N') \mod N \leq M \cdot m_3 < N$$

Note that, in practice, the additional constraint imposed on the domain size N' by Eq. (4.8) does not represent a limitation. For security considerations, the magnitude of modulus N must be at least 768 bits large. Consider values of a and b that can be represented on 64 bits, for instance. Such values are sufficiently large for many applications. In this case, the random blinding factor domain will be bounded by $M = \frac{2^{768}}{2} \cdot \frac{1}{2^{64}}$, which is in the order of 2^{700}, sufficiently large to obtain a strong degree of protection through random blinding.

The protocol for private evaluation of point-rectangle enclosure builds upon the sign evaluation protocol. Denote the user location by coordinates (x_u, y_u), and let the server-stored rectangle R be specified by its lowest-left (LL_x, LL_y) and upper-right (UR_x, UR_y) coordinates. Consider the example in Figure 4.5(a): the user location is situated inside the rectangle if and only if the four inequalities hold simultaneously. Conversely, if any of the inequalities does not hold (Figure 4.5(b)), the user is outside the rectangle (or on the boundary of R).

The enclosure condition can be privately evaluated by running the $sign(b-a)$ protocol for each of the four inequalities.

In practice, spatial coordinates are represented as floating point numbers, either in single (32-bit) or double (64-bit) precision. On the other hand, Paillier encryption requires the use of positive integers alone. Nevertheless, the message space $\mathbb{Z}_N$ is large enough to accommodate even the most demanding application requirements with respect to coordinate precision. During the protocol execution, floating point values are converted to fixed precision. For instance, assume that

the spatial data domain is $[0, 10^6]^2$ and six decimal points are required. Then, $2 \cdot \lfloor \log(10^6) \rfloor = 34$ bits are sufficient for this representation, much lower than the magnitude of N. This leaves a very large domain for the values of the random blinding factors.

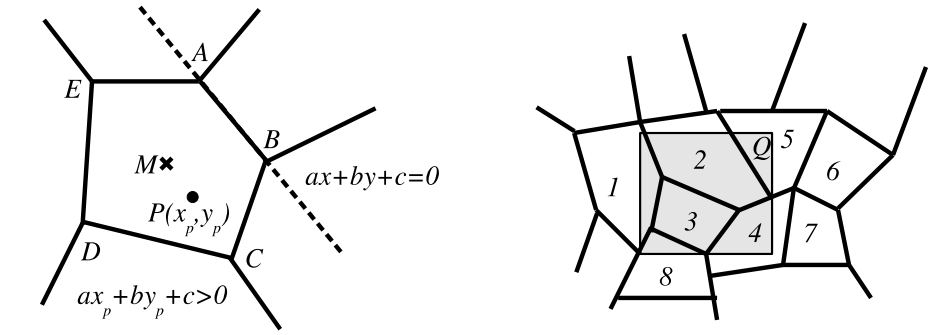
4.4 PRIVATE EVALUATION OF POINT-CONVEX-POLYGON ENCLOSURE

The protocol for private evaluation of point-in-convex-polygon enclosure lies at the foundation of the hybrid technique for answering exact NN queries. The exact hybrid NN technique is ideal from the point of view of data disclosure, as it returns a single POI that is the *exact* NN of the query point. In this case, an optimal outcome is achieved from both the querying user's point of view, who receives his or her exact NN, as well as from the database point of view, since only a single data point is disclosed per query.

We introduced in Chapter 3 the method from [23] which answers private exact NN queries using Voronoi tessellations and PIR. The idea behind that pure-PIR technique is to use a regular 2D grid and to create a bucketing scheme where each cell in the 2D grid is assigned all data points whose Voronoi cells intersect the grid cell. At query time, the client performs a PIR query for the grid cell that encloses his or her location. Although the method guarantees that the exact NN is part of the result received by the user, the bucketization from Voronoi cells to grid cells involves an inherent loss of precision, due to the infeasibility of having a grid fine-grained enough such that only a single Voronoi cell is hashed in each grid cell. In fact, experimental results from [23] with a real-life dataset show that the average number of data points hashed in a grid cell (and hence disclosed in a single query) is 15. This number is far from the optimal 1. Using a more fine-grained 2D grid cell could potentially lower this number, but results into a rather large computational and communication overhead since the computational complexity of PIR is linear to the number of grid cells. Furthermore, if data are skewed, using a finer grid does not necessarily translates into a decrease in disclosed POI.

The hybrid exact NN solution addressed the above-mentioned limitations by allowing the private interactive evaluation of arithmetic conditions using homomorphic encryption. In particular, the user can privately identify the index of the Voronoi cell that the user's location belongs to *without* need for bucketization.

Voronoi cells in two dimensions are convex polygons. Each side of the polygon belongs to a line with equation $ax + by + c = 0$. A well-known procedure from computational geometry [16] states that it can be determined whether a point $P(x_p, y_p)$ is included in a convex polygon by replacing the variables in the line equations of the polygon sides with the point coordinates. Specifically, denote by ℓ the number of polygon sides (which is also the number of vertices) and denote by



(a) Checking Inclusion of a Point in a Convex Polygon

(b) Filtering Voronoi Cells based on query CR

Figure 4.6: Enclosure evaluation and filtering for Voronoi cells.

$$\begin{aligned} a_1x + b_1y + c_1 &= 0 \\ a_2x + b_2y + c_2 &= 0 \\ &\cdots \\ a_\ell x + b_\ell y + c_\ell &= 0 \end{aligned} \tag{4.10}$$

the set of equations corresponding to the polygon sides traversed in clockwise direction along the polygon perimeter. If the sign of all expressions obtained by replacing the coordinates of P in all equations in Eq. (4.10) is positive (or zero), then the point P is inside (or on the edge) of the polygon. Figure 4.6(a) illustrates this concept. For instance, by replacing the coordinates of the mass center M of the polygon in the equations of the individual sides, a non-negative value is always obtained for each expression.

Using this simple procedure in conjunction with homomorphic encryption allows clients to learn the index of the particular Voronoi cell they belong to, without learning either the extent of that cell or the extent of any other Voronoi cell for that matter. In fact, the only information that the user learns is how many Voronoi cells are in the dataset (or in the window corresponding to the query CR). Note that the expressions that need to be evaluated have linear form, hence the algorithms for private evaluation of the sign of sum/difference from Section 4.3 can be reused without change. Similar to the hybrid model used for approximate NN queries, where only intersecting rectangular tiles were included in the computation, the query CR Q is used to filter the Voronoi cells that are candidates for the exact NN result, as shown in Figure 4.6(b). Any existing spatial index structure can be used to efficiently determine matching cells, e.g., R^*-trees.

Note that, Paillier homomorphic encryption operations [46] are rather expensive in terms of computational cost. For the approximate NN algorithm of Section 4.3, only four evaluations (one for

each side of a rectangle) were needed, and the number of rectangles was relatively small compared to the number of points. A Voronoi cell can have a number of sides considerably larger than four. However, the most expensive Paillier operations are encryption and decryption. On the other hand, operations with ciphertexts are relatively inexpensive. As it turns out, the server only needs to perform a single encryption operation per polygon side, for the free term c in the line equation. All other operations are ciphertext-ciphertext multiplications or ciphertext-plaintext exponentiations.

Similar to the approximate NN protocol, assume that real-valued point coordinates are converted to integers (for a large-enough value of N the loss of precision is negligible). The protocol executed by the client and server to privately answer exact NN queries consists of the following steps:

1. The client, situated at location (x_p, y_p), generates a public key E and private key D for Paillier encryption with modulus N. The client sends the server public key E, the CR Q, as well as the ciphertexts $E(x_p)$, $E(y_p)$, $E(N - x_p)$, $E(N - y_p)$ (the need for the latter pair of ciphertexts will become evident in Step 2).

2. For each Voronoi cell v_j that intersects Q, denote by ℓ_j the number of sides of the polygon representing v_j. For every side i, $1 \leq i \leq \ell_j$, with corresponding line equation $a_i^j x + b_i^j y + c_i^j = 0$ the server computes the value

$$M_i^j = E(x_p)^{a_i^j} \times E(y_p)^{b_i^j} \times E(c_i^j)$$

   The value M_i^j corresponds to the ciphertext of the value that indicates if the user coordinates are inside the cell v_j with respect to side i. Only the sign of the value is required to evaluate enclosure, and to protect the spatial extent of the cell from the client, the server blinds the value with a multiplicative random constant $r > 0$:

$$M'^j_i = (M_i^j)^r$$

   All values M'^j_i are returned to the client. Note that, if any of the constants a_i^j or b_i^j are negative, then the above operations are done with respect to the absolute value of these constants, and the ciphertexts $E(N - x_p)$, $E(N - y_p)$ are used instead. This procedure solves the issue of Paillier encryption not supporting directly subtraction (a similar mechanism was explained in detail in Section 4.3 for approximate NN). In addition, to prevent the client from inferring any information about the geometry of adjacent cells based on the number of polygonal sides in consecutive cells, the M' values can be randomly permuted with respect to their j coordinate. In other words, the client receives the ciphertexts of blinded enclosure expressions of cells in random order (although the grouping of polygon sides with respect to each cell is preserved intact).

3. The client decrypts the received ciphertexts, and checks to see which cell j_0 has all values $D(M'^{j^0}_i)$ positive. Note that, as a performance optimization, if at least one M'^j_i value for the

currently considered cell j is negative, the rest of the ciphertexts for that particular cell need not be decrypted, since it is clear that the enclosure condition does not hold for that cell. Finally, after identifying the enclosing Voronoi cell index, the client performs a PIR retrieval for the cell with index j_0. Note that it is guaranteed that the PIR object associated to a Voronoi cell contains a single data point, hence the PIR phase is much more efficient than for either approximate NN or the exact NN protocol from [23], for which there are a large number of data points associated with each PIR matrix item.

At the end of the protocol, the client learns the value of a *single* data point which is the exact NN of the client's location.

CHAPTER 5

Private Matching of Spatial Datasets

Private matching (or join) of spatial datasets has several important practical applications. Various providers of Internet services may gather over time a considerably large subscriber base, and these customers may be interested in receiving service personalized to their geographical location from another provider. A typical case is that of geosocial network providers who have access to location data of large numbers of users. Consider for instance two distinct such service providers, A and B, that have many customers in the same city. Both parties may want to provide improved service to their users, e.g., location-based notification of public events which are recommended by users with similar profiles across networks. Alternatively, a match-making service that takes into account location proximity could also be a motivating application for service providers to share privately information outside their network.

A typical objective in such a scenario would be to evaluate privately the result of a spatial join of the form *"find the pairs of customers of parties A and B that are separated by a distance below 1,000 meters."* The query result reveals to the two parties the sets of users that match the condition. However, in the process of query evaluation, locations of customers that are *not* part of the join result may also be revealed to the other party. Such disclosure is not desirable, since the customer base is an important asset to each company. For instance, party B may act maliciously and sell the information about the locations of A to a competitor. Therefore, it is important to perform the join in a privacy-preserving fashion, in the sense that no additional locations other than the ones included in the join result should be disclosed in the matching process.

5.1 PROBLEM FORMULATION

Consider parties A and B and their respective spatial datasets D_A and D_B. Given a distance threshold δ expressed with respect to the Euclidean distance d_E, the two parties want to determine the result of the join set

$$D_A \bowtie_\delta D_B = \{(p, q) \mid p \in D_A \wedge q \in D_B \wedge d_E(p, q) < \delta\}$$

without disclosing any of their data points that are not part of the join result. Since none of the points of the other party are known in advance, both A and B must include all of their points in the join query processing. To prevent disclosure, they first transform their points. Denote by $D_A \bowtie_\delta D_B$ the *actual* result of the join, and by R the result obtained using the transformed data.

R also includes points that are disclosed in plaintext format due to false positive. There are several important properties that solutions for private matching must satisfy:

- *Robustness against background-knowledge attacks.* Given a transformed data point, an adversary must not be able to identify its original location.
- *Precision.* The joining process must not disclose any original points that are not part of the result. Formally, precision is defined as

$$\frac{|(D_A \bowtie_\delta D_B) \cap R|}{|R|}$$

- *Recall.* The joining procedure must return the complete result. Recall measures the number of missed results, i.e.,

$$\frac{|(D_A \bowtie_\delta D_B) \cap R|}{|D_A \bowtie_\delta D_B|}$$

- *Overestimation Ratio (OR).* For every pair $(p, q) \in D_A \bowtie_\delta D_B$, the distance $d_E(p, q) < \delta$. If a method obtains false positives, then in addition to the number of false positives, it is also important to determine how far apart these points are compared to δ. For instance, if $d_E(p, q) = 110$ meters and the threshold is $\delta = 100$ meters, the disclosure is more acceptable than if $d_E(p, q) = 1{,}000$ meters. OR is defined as

$$OR = \frac{\max_{(p,q)\in R} d_E(p, q)}{\delta}$$

An ideal technique for matching must be robust against reverse-engineering, and must achieve 100% precision and recall (which implies $OR < 1$).

5.2 DATASET MAPPING

The work in [26] introduces a technique for private evaluation of spatial joins using geometric transformations. Data points are mapped from 2D coordinates to a multi-dimensional space. The mapping is specifically designed to protect against adversaries with background knowledge. The key idea behind the transformation is the use of a two-level grid. Each level-1 regular grid splits the data space into a set of *major* cells, and each major cell is further split according to a level-2 grid into a set of *minor* cells. The mapping uses multiple instances of the two-level grid, where each instance represents a rotation of the grid by a random angle around a randomly chosen pivot.

To understand why the two-level grid design is an appealing design choice, consider two attack venues that are representative for adversaries with background knowledge: *distance-based* and *placement-based* attacks. If the geometric transformation of datasets preserves large distances between points, then an attacker can identify certain patterns among the mapped points that allow the isolation of outliers. The first-level grid illustrated in Figure 5.1(a) partitions the data space into

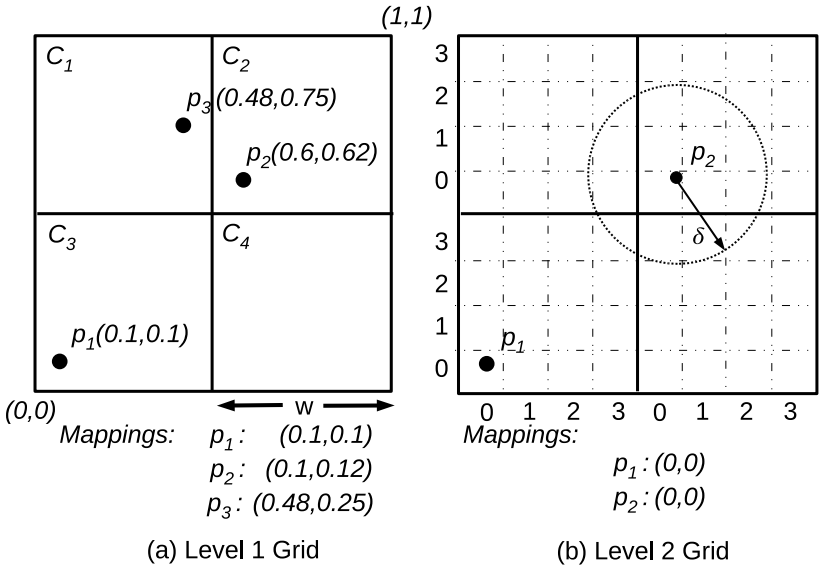


Figure 5.1: Two-level grid design.

a number of major cells of size $w \times w$ each, and the coordinates of each point are considered with respect to the lower-left corner of the enclosing major cell (e.g., the mapping of p_2 corresponds to its coordinates within cell C_2). The major grid reduces the distance between each two points to a maximum of $w\sqrt{2}$, effectively eliminating the notion of outlier from the attacker's view of the data, hence thwarting distance-based attacks. For instance, although points p_1 and p_2 are situated far apart from each other, the distance between their mapped values is small.

While the major-grid split protects against distance-based attacks, it is still vulnerable to another type of attack, namely, placement-based attacks. Revisiting the example in Figure 5.1(a), consider that the attacker has knowledge of the existence and exact coordinates (i.e., placement) of point p_1. Furthermore, assume that p_1 is close to the lower-left corner of the dataspace, hence the attacker can infer with high probability that p_1 is enclosed by the lower-left major-grid cell C_3. In this case, the mapped coordinates of p_1 coincide with its actual coordinates. If the coordinates p_1^x and p_1^y are expressed with high precision, then it is possible that no other point in the dataset has the mapped value equal to that of p_1 in at least one of the x and y coordinates. For instance, neither p_2 nor p_3 have 0.1 in the mapped y value. Therefore, the adversary learns that p_1 must belong to the dataset.

To protect against placement attacks, the exact coordinates of data points should not be disclosed. Instead, a discretization of the data space is performed by dividing each major-grid cell

according to a $k \times k$ minor-grid.[1] Points within the same minor cell are mapped to the same coordinates, representing the position (row and column) of the minor cell inside the major cell. A similar division structure is repeated for all the major cells; consequently all the points enclosed within the same row and column (belonging to the *same* or *different* major cells) share the same combination of coordinates. For instance, in Figure 5.1(b) a 4×4 minor-grid ($k = 4$) is used, and both p_1 and p_2 are mapped to value $(0, 0)$, which corresponds to the coordinates of the lower-left minor-grid cell in each of the corresponding major cells.

Note that the major/minor grid transformation protects the location of points, but it also has a negative impact in the precision of evaluating matching pairs. For instance, points p_1 and p_2 that have the same mapped value in Figure 5.1(b) will be considered as matching, although the Euclidean distance between them is larger than threshold δ because only information about the minor cells, and not the major cells, is employed in the matching decision). This concept can be visualized in Figure 5.2, that illustrates the cells matching with point p for $k = 6$ and two different

[1]Note that, this k value has no relation to the concept of k-anonymity used in previous chapters for LBS protection. A different privacy model is used in this solution, unrelated to k-anonymity.

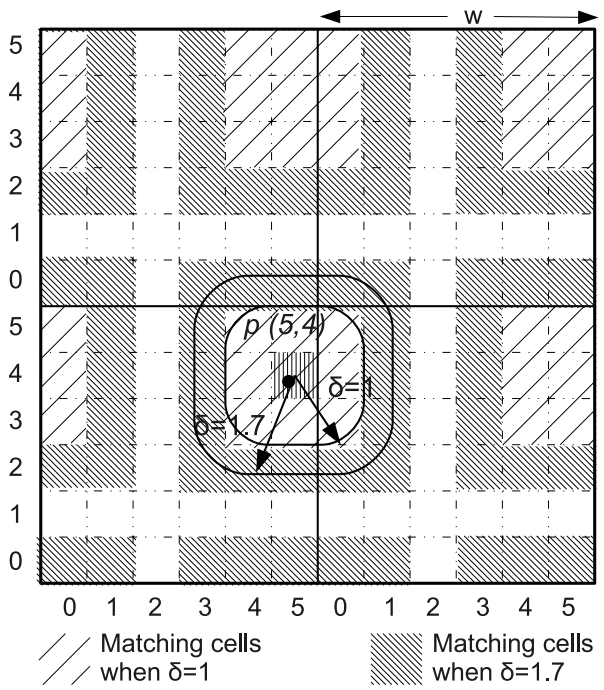


Figure 5.2: Minkowski sum and matching cells.

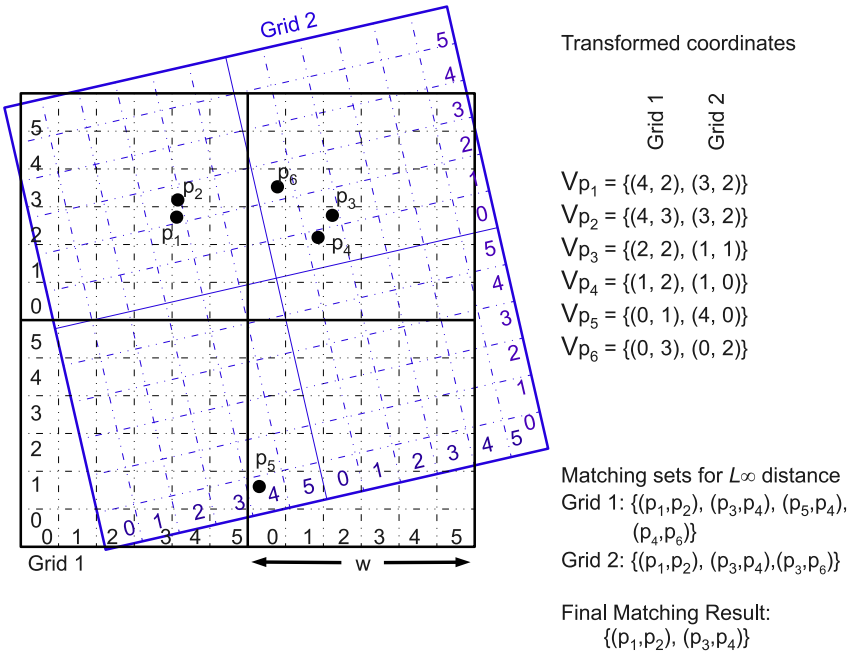


Figure 5.3: Matching with rotated grid instances, $\delta = 1$ (L_∞ metric).

thresholds. First, p is mapped to its minor cell coordinates, i.e., $p(5, 4)$. It can be observed that, for a certain threshold δ, the matching cells are the cells situated at a Euclidean distance $< \delta$ of *any* point within cell $(5, 4)$. This spatial constraint is represented by the Minkowski sum of cell $(5, 4)$ and a circle with $radius = \delta$. The Minkowski sums for two choices of the matching threshold ($\delta = 1$ and $\delta = 1.7$) are shown. For each of them, the shadowed areas mark the cells contained or intersected by the Minkowski sum, which represent the matching cells. Due to the replication of the minor cell coordinates in each major cell, the group of matching cells appears at the same relative position in each major cell.

In order to facilitate pruning of non-matching pairs of points based on their mapped values, several distinct instances of the two-level grid are employed, obtained through rotation by a random angle around a randomly chosen pivot point. Figure 5.3 shows an example with two grids, $k = 6$ (for simplicity, the pivot point is not shown). The mapped value of each point consists of two pairs of coordinates, one pair for each grid (the first two coordinates correspond to the grid with zero-rotation angle). For ease of illustration, consider that the join condition requires that two points match if the L_∞ distance between their mapped coordinates is at most $\delta = 1$ in each grid. There are four matching pairs of points according to $Grid_1$, but only three according to $Grid_2$. It is straightforward to prove that the two-level grid mapping is contractive with respect to the L_∞ metric, therefore

Table 5.1: Summary of notations

Symbol	Description
w	size of major-grid cell
k	granularity of minor-grid ($k \times k$)
δ	matching threshold
r	number of rotations
$u_i = (u_i^x, u_i^y), (1 \leq i \leq r)$	rotation pivots
$\alpha_i, (1 \leq i \leq r)$	rotation angles
$p' = (p'.c_1, \ldots, p'.c_r)$	mapping of data point p

a pair of points can match in the original data space only if *all* their mappings (with respect to each individual grid) match. By using several distinct rotated grids the matching precision can be improved.

The parameter notations are summarized in Table 5.1. The choice of parameters is important with respect to both privacy and matching precision. The primary objective of the two-level grid structure is to prevent disclosure of point locations through distance- and placement-based attacks. However, due to the fact that the distances are transformed, matching precision decreases. The choice of w is particularly important, and it depends on the value of the matching threshold δ. For instance, if $w \leq \delta\sqrt{2}$, no pruning is possible with the two-level grid. However, in general, threshold distances are small compared to the dataspace size (for instance, in a 50×50 km city, it may be interesting to find pairs of points situated at most 500 meters apart from each other, i.e., 1% of the dataspace). Therefore, it is possible to find w values that are several times larger than δ, but still considerably smaller than the data space size.

5.3 JOIN PROCESSING

The system architecture consists of three entities: the data owners A and B, and a semi-trusted third party TP. The TP may try to infer the actual placement of points of A and B, but it is assumed that it will not collude with either of the parties A or B. In the semi-honest trust model, all parties follow the protocol, but they may try to infer additional information.

The private matching process with geometrical transformations consists of four steps, and is illustrated in Figure 5.4.

1. Parties A and B agree on the transformation parameters w, k, r, α_i and u_i $(1 \leq i \leq r)$, as well as matching threshold δ (step 1).

2. Parties A and B transform their datasets according to the chosen parameter set (step 2) and send their mapped data to TP, as well as k and the matching threshold for the transformed space (step 2′).

3. TP processes the spatial join with respect to the mapped datasets (step 3) and returns the matching point identifiers to parties A and B (step 3′).

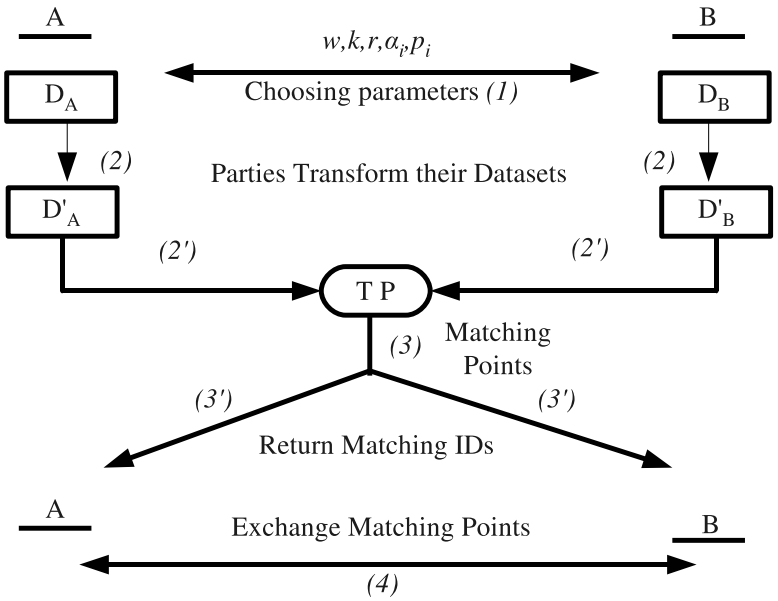


Figure 5.4: System architecture.

4. Parties A and B exchange the actual points corresponding to the matching identifiers (step 4).

The two parties A and B need to transform their data points into vectors that will be processed by the TP in the next step. The transformed vector of a point consists of a list of $2 \times r$ integers corresponding to the coordinates of the minor cell enclosing the point in each of the r grid instances. Each instance corresponds to a rotation of the original data space with angle α_i around a pivot $u_i = (u_i^x, u_i^y)$, where $1 \leq i \leq r$. The resulting mapped value p' is denoted by

$$p' = \left(p'.c_1, p'.c_2, \ldots, p'.c_r\right), \; c_i = (c_i^x, c_i^y), c_i^{x,y} = 0..k-1 \tag{5.1}$$

The complete mapping procedure executed by each party is described in the pseudocode from Figure 5.5.

The TP determines the matching pairs of points in the mapped dataspace, based on the point mappings sent by parties A and B. Note that, in the mapped space, the distance measurement unit is the side length of a minor cell (w/k), hence the matching threshold becomes

$$\delta' = \delta \times k \div w. \tag{5.2}$$

This value is sent to the TP by either A or B, together with their transformed points. Note that neither the original threshold δ, nor the window size w need to be disclosed to the third party. Although the value of k is required for the distance computation, this disclosure does not lead to any

Dataset Mapping
Input: dataset $D, w, k, r, \alpha_i, u_i = (u_i^x, u_i^y)$
Output: transformed dataset D'
1. **for** $i = 1$ **to** r **do**
   /* create transformation matrices */
2. $R_i = \begin{bmatrix} \cos\alpha_i & \sin\alpha_i & 1 \\ -\sin\alpha_i & \cos\alpha_i & 1 \\ 0 & 0 & 1 \end{bmatrix}, T_i = \begin{bmatrix} 1 & 0 & -u_i^x \\ 0 & 1 & -u_i^y \\ 0 & 0 & 1 \end{bmatrix}$
3. **forall** $p \in D$ /* compute mapped vector p' for each p */
4. $p' = \emptyset$
5. **for** $i = 1$ **to** r **do**
6. $[p_i^x \; p_i^y \; 1]^T = T_i^{-1} \times R_i \times T_i \times [p^x \; p^y \; 1]^T$
7. $c_i^x = \lfloor (p_i^x - w \cdot \lfloor p_i^x / w \rfloor) \cdot k / w \rfloor$ /* x-minor cell */
8. $c_i^y = \lfloor (p_i^y - w \cdot \lfloor p_i^y / w \rfloor) \cdot k / w \rfloor$ /* y-minor cell */
9. $p' = p' || (c_i^x, c_i^y)$
10. $D' = D' \cup p'$

Figure 5.5: Dataset transformation.

privacy leakage as this value can be learned by looking at the coordinates c_i^x, c_i^y of the transformed vectors.

The matching function for every candidate pair of vectors ($p'_A \in D'_A$, $p'_B \in D'_B$) computes the distance $cdist$ between the cells of p'_A and p'_B for each rotated grid instance, and compares it to δ'. The pair (p'_A, p'_B) represents a match only if $cdist < \delta', \forall i = 1..r$, i.e., for all rotated grid instances. Figure 5.6 shows the algorithm executed by the TP.

Matching
Input: transformed datasets D'_A, D'_B, matching threshold δ'
Output: set of matching pairs S
1. $S = \emptyset$
2. **forall** $p'_A \in D'_A$
3. **forall** $p'_B \in D'_B$
4. **for** $i = 1$ **to** r **do** /* do matching for every rotation */
5. $c_A = (p'_A.c_i)$ /* minor cell of p'_A for rotation r */
6. $c_B = (p'_B.c_i)$ /* minor cell of p'_B for rotation r */
7. **if** $cdist(c_A, c_B) \geq \delta'$ /*cell distance*/
8. **break** /* discard pair */
9. **if** $i = r$ /* if we reached the last rotation */
10. $S = S \cup (p'_A, p'_B)$ /* add pair to the result */

Figure 5.6: Matching in the third party.

The function $cdist(c_A, c_B)$ computes the distance between the two minor cells c_A and c_B, defined as the Euclidean distance between the two closest points of c_A and c_B. Note that two cases

must be considered: *(i)* c_A and c_B belong to the same major cell, or *(ii)* they belong to distinct major cells. For the first case, $cdist$ is given by

$$cdist(c_A, c_B) = \sqrt{\Delta_x^2 + \Delta_y^2} \tag{5.3}$$

where

$$\Delta_x = \begin{cases} 0, & c_A^x = c_B^x \\ |c_A^x - c_B^x| - 1, & otherwise \end{cases}$$

and similarly for the y coordinate. Figure 5.7 shows how this distance is computed: in major cell C_0, the distance between p_A and p_B is given by $d_0 = \sqrt{2^2 + 3^2} = \sqrt{13}$.

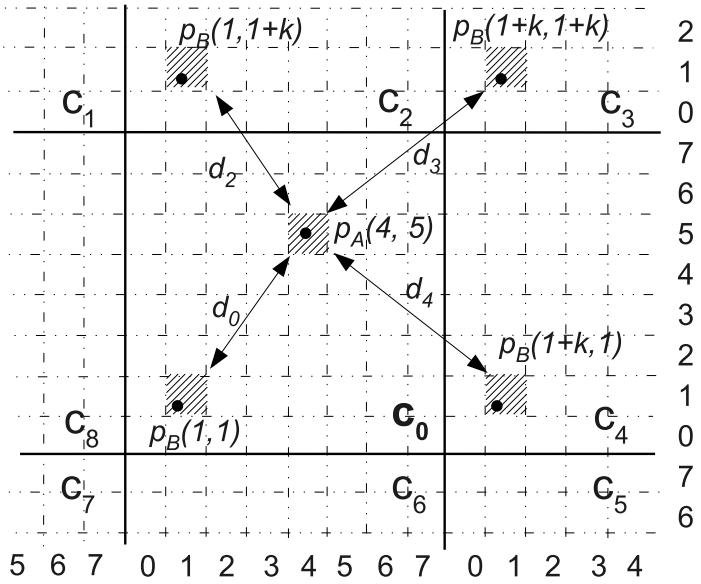
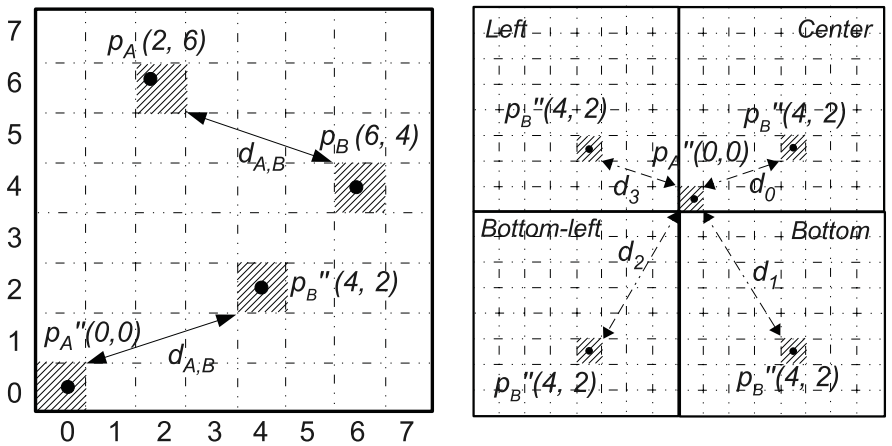


Figure 5.7: Euclidean distance calculation between grid cells.

For the second case (when the points belong to different major cells) we must consider the minimum possible distance between them. In particular, since we are interested in the minimum distance, we only need to consider combinations of major cells that are neighbors (i.e., major cells C_0 to C_8 in Figure 5.7). Point p_A is marked only in the major grid cell C_0, whereas the placement of p_B is illustrated for multiple major cells. Formally,

$$\begin{aligned} cdist(c_A, c_B) &= \min_{i,j} cdist(c_A, c'), \\ where \quad & c' = (c_B^x + i, c_B^y + j), \\ & i \in \{-k, 0, k\}, j \in \{-k, 0, k\} \end{aligned} \tag{5.4}$$

Note that this definition of distance is symmetric: $cdist(c_A, c_B) = cdist(c_B, c_A)$. In addition, due to symmetry across major cells, we can compute $cdist(c_A, c_B)$ by translating the two minor cells of points p_A and p_B such that c_A is moved to the origin (cell 0, 0) and c_B is moved to the cell $(|c_A^x - c_B^x|, |c_A^y - c_B^y|)$. This property is shown in Figure 5.8(a), where the distance between cell $c_A(2, 6)$ and $c_B(6, 4)$ is equivalent to the distance between $c_A''(0, 0)$ and $c_B''(4, 2)$. Then, the computation in Eq.(5.4) only needs to be performed for the *left*, *bottom-left*, *bottom,* and *center* major cells (see Figure 5.8(b)). This optimization reduces the number of distance evaluations from nine to four.



(a) Euclidean distance between minor cells

(b) Euclidean distance between major cells

Figure 5.8: Optimization of distance calculation.

5.4 ENHANCING PRIVACY BY USE OF CHEBYSHEV DISTANCE

The method described in the previous section requires the disclosure of minor cells coordinate pairs (c_x, c_y) for each point, which may help an adversary to learn the exact placement of the point within its major cell. Since the actual major cell is not disclosed, the attacker does not gain significant knowledge of the position of the point in the data space, especially if w is small. Nevertheless, the privacy level attained can be improved by breaking the correlation between the c_x and c_y cell coordinates of each mapped point, as discussed next.

Consider the Chebyshev distance (L_∞) between a pair of cells, which is defined as the *maximum* of their differences along any of the coordinates x and y. If a pair is a match, then the distance must be below a threshold for *both* coordinates, and therefore the evaluation can be done sepa-

rately for each dimension. Furthermore, the Chebyshev distance is an upper bound of the Euclidean distance and therefore preserves the contractiveness property (i.e., no false negatives can occur).

When using Chebyshev distance, the transformed matching threshold changes to:

$$\delta_c = \lceil \delta \times k \div w \rceil = \lceil \delta' \rceil, \tag{5.5}$$

which is equivalent to aligning δ' to a multiple of minor grid cell side length. The utilization of a coarser distance approximation may lead to decreased precision.

The distance function between minor cells changes compared to the case of Euclidean distance. Specifically, if two minor cells are placed in the same major cell, we have

$$cdist(c_A, c_B) = \max(|c_A^x - c_B^x|, |c_A^y - c_B^y|) \tag{5.6}$$

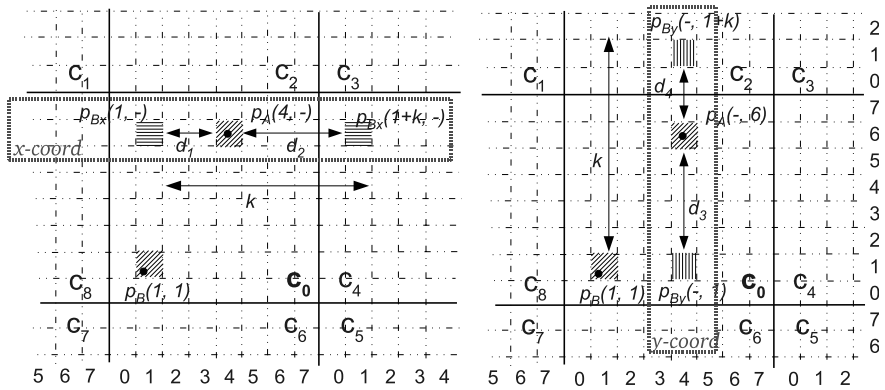


Figure 5.9: Chebyshev distance calculation across distinct major cells.

The distance evaluation for all possible major cell placements is:

$$cdist^x(c_A, c_B) = min(\left|c_A^x - c_B^x\right|, k - \left|c_A^x - c_B^x\right|) \tag{5.7}$$
$$cdist^y(c_A, c_B) = min(\left|c_A^y - c_B^y\right|, k - \left|c_A^y - c_B^y\right|) \tag{5.8}$$

The term $k - \left|c_A^x - c_B^x\right|$ (or $k - \left|c_A^y - c_B^y\right|$ for the y coordinate) reflects the case where the coordinates belong to neighbor major cells. Recall that the minimum distance between any pair of cells is always located in one of the following cases: *(i)* the cells belong to the same major cell, or *(ii)* the cells belong to neighbor major cells. For the latter, the distance between two neighbor cells in one dimension is at most k, therefore the difference for the coordinates becomes $k - \left|c_A^y - c_B^y\right|$. Figure 5.9 illustrates this concept for the distance between $p_A(4, 6)$ and $p_B(1, 1)$ for the two coordinates separately, with $k = 8$. For instance, for the y coordinate $cdist^y(c_A, c_B) = min(|6 - 1|, 8 - |6 - 1|) = min(5, 3) = 3$, and similarly for the x coordinate.

The matching algorithm executed in the third party is similar to the one in Figure 5.6, except that the values retrieved in lines 5–6 are single coordinates (as opposed to cell coordinates). Therefore, the distance computation in line 7 is modified to account for Chebyshev distance, and the *for* loop in line 4 is executed for twice as many iterations, one per each separate coordinate x and y.

Note that, since the two components of the Chebyshev distance (x and y) can be evaluated independently, the TP no longer requires a particular order of the coordinates in the multi-dimensional vector representing a point mapping (Eq.(5.1)). Therefore, parties A and B can agree on a random permutation used by both when arranging the cell coordinates of the same point for distinct grid instances. This prevents the TP from correlating x and y coordinates of a point within the same rotated grid.

CHAPTER 6

Trajectory Anonymization

Figure 6.1 shows the typical scenario of trajectory anonymization: a central trusted component (e.g., telecom company) gathers location samples from a large number of subscribers. The collected data is then shared with other un-trusted entities for various purposes, such as traffic optimization research. The trusted entity is the data *publisher*, and must ensure that releasing the data does not compromise user privacy. The publisher is assumed to be bound by contractual obligations to protect the users' interests, therefore it is trusted not to allow privacy breaches to occur. Location collection is performed in an on-line fashion. Anonymization, on the other hand, is likely to be a more costly operation, and can be performed off-line (although on-line anonymization is also possible).

When trajectory[1] data is published, a malicious attacker may associate user identities with sensitive locations, thus compromising users' privacy. One straightforward solution to protect privacy is to remove all user identifiers from the data, or replace them with pseudo-identifiers. The work in [8] proposes the use of *mix zones*, i.e., areas with high user density where many paths intersect. The idea is to confuse adversaries by inter-changing pseudo-identifiers among users within the mix zones. However, such an approach is not sufficient against sophisticated adversaries who may possess background knowledge (e.g., address of a user's residence) and may employ advanced target-tracking algorithms [47] to re-construct trajectories from successive location samples.

Privacy-preserving trajectory publication techniques can be broadly classified into two categories: methods that publish independent location samples, and methods that publish individual trajectories. Techniques in the former class are suitable for applications where only aggregate location data is required, e.g., traffic monitoring. Such methods are reviewed in Section 6.1. Solutions in the latter category publish individual trajectories, but distort location samples at each timestamp. These methods are reviewed in Section 6.2, and are suitable for applications where the causality relationship between source and destination locations is important.

6.1 PUBLISHING INDEPENDENT LOCATION SAMPLES

The early work of [28] proposes three algorithms to prevent disclosure of sensitive location data. All three assume the existence of a map with the sensitive locations in the enclosing geographical region. The first solution, called *base*, simply suppresses from publication those updates corresponding to sensitive locations. This method achieves only weak privacy. The second solution, called *bounded-rate*, aims to achieve strong privacy by suppressing with fixed frequency updates in non-sensitive areas (in addition to withholding sensitive locations). The idea is to filter out some of the non-sensitive

[1] We use the terms *trajectory* and *track* interchangeably.

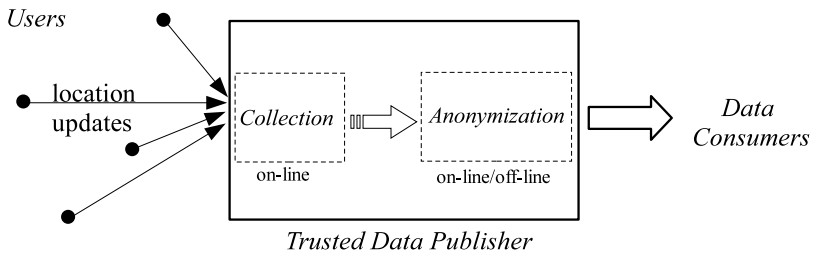


Figure 6.1: Trajectory publication: system architecture.

locations that may represent entry points to sensitive areas. Finally, the third method, called *k-area*, splits the map into zones, or cells, that have both sensitive and non-sensitive locations. All updates in a given cell are delayed until the user exits that particular cell. Subsequently, the previous cell locations are disclosed only if the user had not visited any sensitive location in that cell. None of these three solutions protects against an attacker with background knowledge.

Later in [30], the same authors show that multiple-hypotheses tracking algorithms (MTT) [47] based on Kalman filters are very successful in re-constructing trajectories (in some cases, by matching close to 100% of location samples to the correct path). Once the trajectories are re-constructed, they can be mapped to user identities, based on a small number of known samples (e.g., a user's home or office). In addition, the success probability of identifying tracks and matching users to tracks increases with the duration of track acquisition. Figure 6.2(a) shows an example with two user trajectories $A : a_1 - a_2 - a_3$ and $B : b_1 - b_2 - b_3$. Even if user identifiers are suppressed, and location samples are independently published for each timestamp (i.e., as pairs $(a_1, b_1) \cdots (a_3, b_3)$), filtering allows an attacker to assign sample a_3 to track A with 90% probability. To confuse the attacker, individual location samples are distorted, in order to minimize the probability of successfully matching locations to trajectories. For instance, in Figure 6.2(b), the locations of samples a_3 and b_3 are changed to a_3' and b_3', respectively. The attacker assigns location a_3' to track B with high probability (0.8). Therefore, track matching for the third timestamp (and likely subsequent timestamps as well) is prevented.

Distorting location samples inherently introduces data inaccuracy, and may impact correct query processing on top of the data. A trade-off among privacy and accuracy emerges. The authors of [30] propose metrics to quantify both privacy and inaccuracy. Specifically, privacy is measured by the *expectation of distance error*, which captures how accurate an adversary can match locations to tracks. Given N users (hence N location samples at each timestamp) and an observation time of M timestamps, expectation of distance error for trajectory of user u is measured as

$$E[u] = \frac{1}{NM} \sum_{i=1}^{M} \sum_{j=1}^{I_i} p_j(i) d_j(i)$$

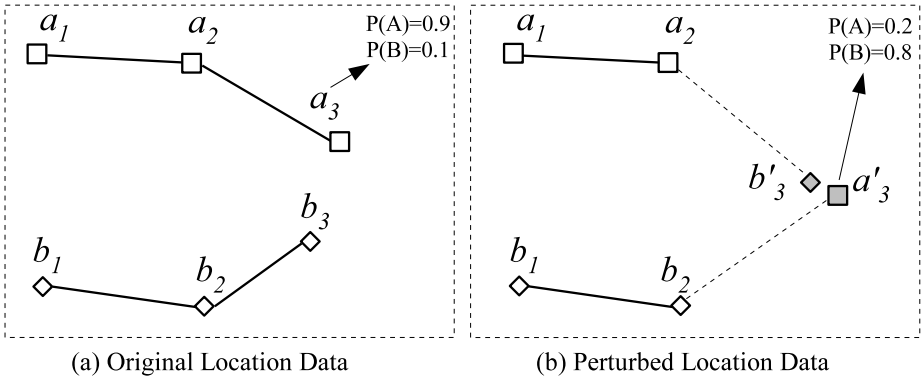


Figure 6.2: Path perturbation prevents trajectory re-construction.

where I_i is the total number of assignment hypotheses for user u at timestamp i, $p_j(i)$ is the probability associated with hypothesis j at timestamp i, and $d_j(i)$ is the distance between the actual and estimated position of u at timestamp i. The data accuracy is measured according to the quality of service (QoS) metric

$$QoS = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \sqrt{(x_{u_i}(j) - x'_{u_i}(j))^2 + (y_{u_i}(j) - y'_{u_i}(j))^2}$$

where (x_{u_i}, y_{u_i}) and (x'_{u_i}, y'_{u_i}) are the actual and perturbed coordinates of u_i, respectively.

The data undergo a *Path Perturbation* phase, which formulates the problem of confusing the attacker as a constrained non-linear optimization problem. The objective is to maximize the privacy function E under the constraint that the maximum distortion for each individual published location does not exceed a threshold R, which is application-dependent. The perturbation phase needs to consider all permutations of assigning location samples to tracks, hence the computational cost is very high. For instance to perform perturbation for N user trajectories of M samples each, the complexity is $O(N!)^M$, which is not feasible in practice. To decrease the overhead, a *Path Segmentation* phase is performed prior to perturbation. The idea is to reduce the search space for the constraint optimization problem by pruning some of the most unlikely hypotheses of assigning samples to tracks.

The Path Perturbation algorithm maximizes the privacy metric under the given accuracy constraint R. However, this is not sufficient to protect the privacy of users in sparse areas. If user trajectories are situated far apart from each other, even the best achievable expectation of distance error may not be enough to prevent re-identification. The work in [32] acknowledges this limitation and proposes a solution that relies on the k-anonymity concept. However, the authors observe that simple location cloaking, as used for private queries, is not suitable for publishing trajectory data, because it severely distorts data. In traffic monitoring applications, for instance, in order to match a

user location to a road segment with high probability, the spatial accuracy must be within $100m$. It is shown experimentally that k-anonymization-based spatial cloaking fails to achieve this threshold: for instance, with a real trajectory trace and a relatively low anonymity degree ($k = 3$), the obtained accuracy is $500m$.

In practice, the privacy threat occurs when individual trajectories can be associated with particular users. Furthermore, such association cannot be performed in very dense areas, but only in sparse areas, and the attacker's success probability increases with the length of the disclosed trajectory. Based on these observations, the work in [32] introduces two new privacy metrics. The first one is called *Time-to-Confusion (TTC)*, and measures the maximum number of consecutive timestamps for which locations along the same trajectory are safe to disclose. If TTC is exceeded for a certain track, its subsequent location samples are suppressed, to prevent trajectory re-construction. The second metric used is *tracking uncertainty*, which measures for each user u the entropy

$$H(u) = -\sum_i p_i \log p_i$$

where p_i is the probability of associating u with the location sample i in a particular snapshot. The attacker's *confidence* is $1 - H$. An algorithm is proposed which verifies at each timestamp if the TTC and the attacker's confidence are below certain thresholds. If thresholds are exceeded, the corresponding location samples are suppressed. The work in [31] follows a similar approach, and extends the solution in [32] by allowing location updates only at well-specified points along a *Virtual Trip Line (VTL)*. The points along each VTL are chosen in such a manner that they correspond to privacy-insensitive locations only.

6.2 PUBLISHING INDIVIDUAL TRAJECTORIES

The objective of the techniques presented in the previous section is to prevent an attacker from re-constructing trajectories based on independent locations. Publishing independent location samples is useful for applications that require only aggregate information, such as traffic monitoring. However, in other classes of applications, the movement patterns and the causality relationship between certain source and destination locations may be of interest. In such cases, it is necessary to publish in a privacy-preserving manner entire trajectories, rather than independent location samples. Preventing a malicious adversary from correlating location samples is no longer a challenge. Instead, the focus is on perturbing trajectory data to prevent the association of individuals with trajectories.

The work in [54] considers a scenario where location samples are drawn from a discrete set (e.g., retail points, tourist attractions, etc.), and assumes an attack model with clearly defined background knowledge. Specifically, the attacker already knows some trajectory fragments, and the identity of users corresponding to those fragments. Consider the example of a company P (publisher) that commercializes cards as a convenient form of payment. Such cards can be used to pay for transportation, as well as for day-to-day purchases. In time, P (which is trusted by all card users) will gather large amounts of trajectory data, which can be useful for a variety of purposes (e.g.,

Original Data

t_1	a_1-a_2-a_3-b_1
t_2	a_1-a_2-b_2
t_3	a_1-a_3-b_1
t_4	a_1-a_3-b_2

A's View

u_1	a_1-a_2-a_3
u_2	a_1-a_2
u_3	a_1-a_3
u_4	a_1-a_3

Transformed Data

t'_1	a_1-a_2-b_1
t_2	a_1-a_2-b_2
t_3	a_1-a_3-b_1
t_4	a_1-a_3-b_2

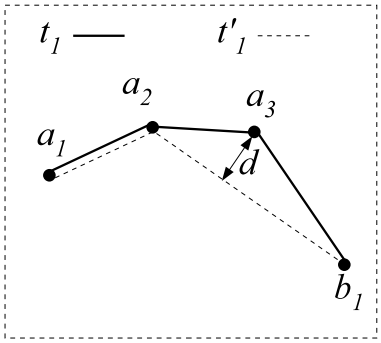


Figure 6.3: Protecting against attackers that know trajectory sub-sequences.

inferring consumer travel and spending patterns). However, P is required by law not to compromise the privacy of its customers. Furthermore, the partner companies of P are not trusted. For example, a retail chain company A has access to all purchases by user u, and also learns the identity of u through a customer-fidelity program. Thus, A has access to a subset of the trajectories followed by u, and may wish to find out what other locations u has visited. P must prevent against this sort of privacy threat.

Consider the example of Figure 6.3, where P publishes the original location data. Partner companies A and B have knowledge about subsets of trajectories corresponding to their points of operation. Such known locations are denoted by a_i and b_j, respectively. A may try to infer the other locations that its customers have visited, by inspecting the original data. For instance, A can learn that u_1 corresponds to trajectory t_1, since only t_1 matches the $a_1 - a_2 - a_3$ movement pattern known by A. Therefore, A can infer with certainty that u has visited location b_1, which may correspond to a nightclub. Such a sensitive association is clearly a privacy breach. The fact that allowed the staging of a successful attack was A's ability to identify the trajectory of u_1. This situation is similar to that of publishing microdata, such as hospital records [48, 51]. In trajectory publication, the location data known to the attacker (A's view) is used as a quasi-identifier, whereas the location samples corresponding to other companies (e.g., B) are sensitive attributes. The authors of [54] approach the trajectory anonymization problem from a data mining [29] perspective. Specifically, each trajectory is regarded as a *transaction*, and each location sample represents a transaction *item*, according to data mining terminology. A particular *itemset* that is present in the attacker's database can be used as a quasi-identifier. To prevent re-identification of trajectories, the published data must consist of transactions in which every itemset occurs a sufficient number of times. In other words, the *support* [29] of each itemset must be large enough to ensure that the privacy breach probability P_{br} is below a threshold. Privacy breach is quantified as the probability of associating an individual trajectory

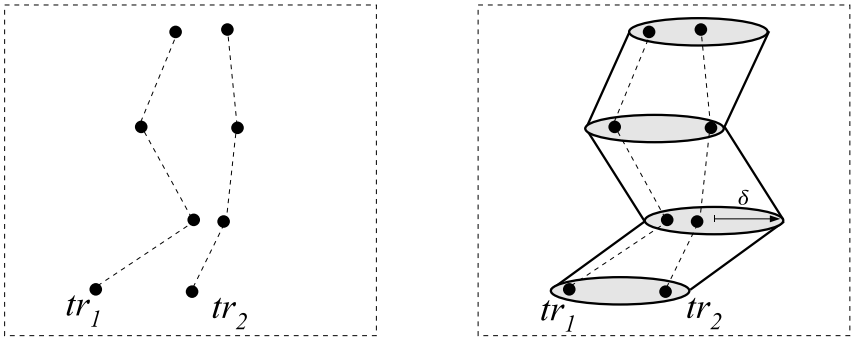


Figure 6.4: Anonymizing trajectories through space translation.

with a particular sensitive location/item, e.g., b_1 in the earlier example. Before publication, the data is sanitized through a greedy heuristic that suppresses individual location samples. In Figure 6.3, the value a_3 is removed from the original data. As a result, A can only infer that the trajectory of u_1 is other t_1' or t_2. Therefore, u_1 may have visited either b_1 or b_2 with equal likelihood, hence the breach probability is reduced from 100% to 50%. The inherent inaccuracy in removing samples is measured by the path deviation introduced. In our example, by removing a_3, the obtained deviation is $dist(t_1, t_1') = d$.

The work in [2] also employs the concept of k-anonymity for trajectories. However, as opposed to [54], it considers continuous-space location samples. Privacy is mainly enforced through location *generalization*, rather than suppression as in the case of [54]. The privacy paradigm proposed by [2] is (k, δ)-anonymity, which is achieved using the concept of *anonymizing tubes*, i.e., a sequence of cylindrical volumes that spatially enclose user trajectories. Figure 6.4 shows an example of $k = 2$ *co-located* trajectories, i.e., trajectories that can be enclosed by a tube with radius δ. It is considered that trajectories are defined over the same discrete time domain $[t_1 \cdots t_n]$, and $\forall t \in [t_1 \cdots t_n]$ and any two trajectories τ_1, τ_2 inside an anonymized tube, it holds that

$$dist((\tau_1[t].x, \tau_1[t].y)(\tau_2[t].x, \tau_2[t].y)) \leq \delta$$

where $\tau[t].x$ and $\tau[t].y$ represent the coordinates of the location sample along trajectory τ at time t.

For each group of trajectories, the polygonal line that represents the center of the tube is published. The data distortion is measured by summing over all timestamps the distance between the location sample of each trajectory and the cylinder center at that particular timestamp. Anonymization is performed through a two-step algorithm. First, a pre-processing phase is employed, to horizontally partition the set of trajectories according to their timeframes (i.e., obtain disjoint sets of trajectories

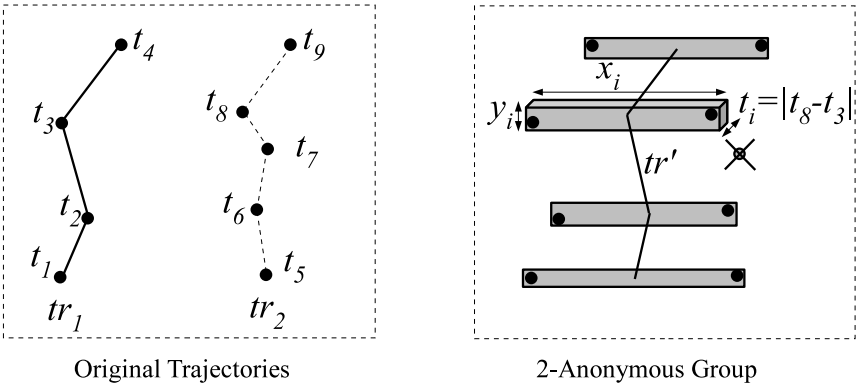


Figure 6.5: Anonymizing trajectories through spatio-temporal generalization.

that are concomitant). Next, a greedy clustering heuristic is executed within each partition, to obtain groups of at least k trajectories each. The tube that encloses each group is published.

In [50], trajectory privacy is also achieved by employing k-anonymity. However, as opposed to [2], trajectories from distinct timeframes can be anonymized together, and their time difference is factored in the data inaccuracy metric used. An algorithm for trajectory clustering in the three-dimensional space-time domain is proposed, which creates groups of at least k transactions each. Trajectories belonging to the same group are generalized, such that they are indistinguishable from each other. Grouping involves generalization of both spatial and temporal coordinates, i.e., all trajectories in the same group are replaced with their spatio-temporal bounding box. An example of anonymizing two trajectories, tr_1 and tr_2, is shown in Figure 6.5. Each location sample is tagged with the timestamp at which it was collected. The resulting bounding boxes together with their time differences (i.e., the time period covered by each bounding box) incur information loss, which is measured according to a *log-cost metric (LCM)*. LCM quantifies the trajectory inaccuracy in both space and time, and is calculated by summing the enlargement required over each of the M published snapshots of locations. A weighting factor for space (w_s), as well as time (w_t) can be specified, depending on the application that uses the data. Formally,

$$LCM = \sum_{i=1}^{M} [w_s(\log |x_i| + \log |y_i|) + w_t \log |t_i|]$$

Anonymization of trajectories is performed in two stages. First, the algorithm chooses the trajectories that will belong to each group. This phase is performed through a heuristic that is similar to string matching. Next, an anonymization phase is performed, where it is decided which samples from each trajectory will be anonymized with samples from other trajectories. Note that

not all trajectories have the same number of samples. Furthermore, not all location samples must be retained: for instance, in Figure 6.5, the sample with timestamp t_7 from tr_2 is suppressed.

CHAPTER 7

Differentially Private Publication of Spatial Datasets

Recently, the *differential privacy* [17] paradigm emerged as a preferred alternative to existing data sanitization techniques. The main benefit of differential privacy is that it provides a *semantic* privacy model with strong protection guarantees that are rooted in formal statistical analysis results. In contrast, previous data sanitization techniques such as generalization techniques (k-anonymity [51], ℓ-diversity [40], t-closeness [39]) only provide a *syntactic* model, i.e., they mandate how the published data should look but are not able to capture the amount of disclosure that occurs due to publication. In particular, syntactic models are notoriously poor in protecting against adversaries with access to background knowledge.

In the realm of spatial datasets, the existence of background knowledge is especially a concern, since large amounts of public information exists in the form of maps, characteristics of public transportation networks, e.g., certain traffic segments are uni-directional, different road segments such as side streets and highway have a large and well-known variance of velocities, etc. The semantic model of differential privacy alleviates all the above concerns related to background knowledge, and allows the data publisher to ensure that an adversary is not able to decide whether a particular individual is included or not in the published dataset, regardless of the amount of additional information available to the adversary.

In this chapter, we review two techniques that allow publication of location information in the context of differential privacy. First, Section 7.1 provides a primer that introduces the reader to the fundamental concepts and principles of differential privacy. Next, in Section 7.2 we look at how differential privacy can be used to publish static datasets indexed in a hierarchical structure. Hierarchical indexes such as quad-trees and kd-trees [16] are very widespread in geospatial applications, and they can serve as a basis for publication of statistical information about datasets of geographical coordinates. However, as shown in [14], the algorithm for building such structures must be modified according to the principles of differential privacy, in order to guarantee that no disclosure of information about individual locations occurs.

Finally, in Section 7.3 we review the work in [10] which provides a method for privacy-preserving publication of trajectories. Note that this is an inherently more complex and difficult problem than the one studied in [14], due to the fact that multiple locations at consecutive timestamps belong to the same individual in the initial dataset of trajectories. Therefore, an adversary is provided with more powerful means of inference by publishing sequencing information as well. Differentially

private trajectory publication is possible through the use of another hierarchical structure in the form of a prefix tree.

7.1 A PRIMER ON DIFFERENTIAL PRIVACY

Instead of publishing perturbed versions of a dataset, differential privacy allows users to interact with the database only by means of statistical queries. Random noise is added to each query result to preserve data privacy. The work in [17, 18] explores differential privacy in a thorough theoretical framework, and shows that the privacy guarantees provided are very strong. Specifically, an adversary that attempts to attack the privacy of some individual entity r will not be able to distinguish from the interaction with the database (called a *transcript*) whether a record representing r is present or not in the database.

A statistical database answers aggregate queries such as *Count* or *Sum* queries. Most works on differential privacy consider aggregate *range* queries, that can be expressed by d-dimensional hyper-rectangles in the attribute domain space $A_1 \times \ldots \times A_d$. A range query is represented as a Cartesian product $Q = [x_1^Q, y_1^Q] \times \ldots \times [x_d^Q, y_d^Q]$ where $[x_i^Q, y_i^Q]$ is the extent of Q on attribute A_i. For instance, in context of a relational database table containing information about the age and salary of individuals, the *Count* query

$$\texttt{SELECT COUNT}(*)\ \texttt{FROM T WHERE}\ (40 \leq Age \leq 60)\ \texttt{AND}\ (30k \leq Salary \leq 40k);$$

has extent [40, 60] on attribute *Age*, and [30k, 40k] on the *Salary* attribute. Such aggregate range queries are also very frequent in spatial databases, where one is interested in finding how many locations reside in a particular rectangular area given as argument to the query.

Statistical databases allow users to retrieve coarse-grained aggregate information. On the other hand, it is important to preserve the privacy of individual records, by not allowing fine-grained aggregate queries. One solution to preserve privacy is to disallow queries that have extents smaller than a certain threshold, e.g., 10 years of age, and 10k salary. However, this may not be sufficient. For instance, consider malicious user Mallory who knows that his colleague Alice is the only one in the company who is 51 years old. Mallory can issue the following two queries:

$$\begin{aligned}
&\texttt{Q1}: \quad \texttt{SELECT COUNT}(*)\ \texttt{FROM T} \\
&\qquad\quad \texttt{WHERE}\ (40 \leq Age \leq 50)\ \texttt{AND}\ (30k \leq Salary \leq 40k); \\
&\texttt{Q2}: \quad \texttt{SELECT COUNT}(*)\ \texttt{FROM T} \\
&\qquad\quad \texttt{WHERE}\ (40 \leq Age \leq 51)\ \texttt{AND}\ (30k \leq Salary \leq 40k).
\end{aligned}$$

Assume that the response to Q_1 is 10, and the response to Q_2 is 11. Then, Mallory can infer that Alice must be included in the result of query Q_2, and hence her salary is in the range [30k, 40k]. Note that, the attack was successful because Mallory was able to access the query results for the two views T_1 and T_2 of table T that differ in a single record (the record of Alice).

Differential privacy [18] aims to prevent this type of inference, by adding random noise to each query result. The interaction between a user and the database is expressed as a *transcript*. Formally,

Definition 7.1 Transcript Let $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$ be a set of aggregate queries, and denote by $Q_{i\ (1 \le i \le q)}^{T}$ the result of answering query Q_i on table T. A *transcript*

$$\mathcal{TR}_{\mathcal{Q}}^{T} = \{(Q_1, a_1), \ldots, (Q_q, a_q)\}$$

is the response of a statistical database to the query set $\mathcal{Q}$ on database table T, where $a_i = Q_i^T$.

A statistical database satisfies ϵ-differential privacy if the ϵ-indistinguishability condition [18] is fulfilled:

Definition 7.2 ϵ-indistinguishability Consider a statistical database that produces the transcript $\mathcal{U}$ on the set of queries $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$, and let $\epsilon > 0$ be an arbitrarily small real constant. Transcript $\mathcal{U}$ satisfies *ϵ-indistinguishability* if for every pair of views T_1, T_2 such that $|T_1| = |T_2|$ and T_1, T_2 differ in only one record, it holds that

$$\left| \ln \frac{Pr[\mathcal{TR}_{\mathcal{Q}}^{T_1} = \mathcal{U}]}{Pr[\mathcal{TR}_{\mathcal{Q}}^{T_2} = \mathcal{U}]} \right| \le \epsilon \tag{7.1}$$

In other words, an attacker is not able to learn whether the transcript was obtained by answering the query set $\mathcal{Q}$ on view T_1, or on view T_2. To achieve ϵ-indistinguishability, a statistical database injects noise into each query result Q_i^T. The amount of noise required is proportional to the *sensitivity* of the query set $\mathcal{Q}$, which is defined as follows:

Definition 7.3 L_1-sensitivity Over any two views T_1, T_2 such that $|T_1| = |T_2|$ and T_1, T_2 differ in only one record, the *L_1-sensitivity* of query set $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$ is measured as

$$S_{L_1}(\mathcal{Q}) = \max_{\forall T_1, T_2} \sum_{i=1}^{q} |Q_i^{T_1} - Q_i^{T_2}|$$

The following theorem from [18] gives a sufficient condition for a statistical database to satisfy ϵ-differential privacy:

Theorem 7.4 *Let $\mathcal{Q}$ be a set of queries and denote by $S_{L_1}(\mathcal{Q})$ the L_1-sensitivity of $\mathcal{Q}$. Then, differential privacy with parameter ϵ can be achieved by adding to each query result random noise X, i.e., $Q_i^T \leftarrow Q_i^T + X$, where X is a random, i.i.d. variable drawn from a Laplace distribution with mean 0 and magnitude $\lambda \ge S_{L_1}(\mathcal{Q})/\epsilon$.*

An essential operation in enforcing differential privacy is determining the sensitivity $S_{L_1}(\mathcal{Q})$. Interestingly, it is shown in [18] that $S_{L_1}(\mathcal{Q})$ is independent of the dataset T, and can be determined based on the query set $\mathcal{Q}$ alone. However, for arbitrary query sets, it is shown in [55] that computing sensitivity is NP-hard. Nevertheless, if certain restrictions are imposed on $\mathcal{Q}$, sensitivity (or its approximation) for sets of *Count* queries can be efficiently computed. Specifically:

(i) If all queries in $\mathcal{Q}$ have disjoint ranges, $S_{L_1}(\mathcal{Q}) = 2$.

(ii) If queries in $\mathcal{Q}$ have overlapping ranges, then a 2-approximation of $S_{L_1}(\mathcal{Q})$ is given by the maximum number of queries that overlap the same point in the data space. Formally, for any data space point $p \in A_1 \times \ldots \times A_d$, define

$$\mathcal{Q}^p = \{Q \in \mathcal{Q} | p \in Q\}$$

i.e., the set of queries that cover point p. Then, we have

$$S_{L_1}(\mathcal{Q}) \leq 2 \cdot \max_{p \in A_1 \times \ldots \times A_d} |\mathcal{Q}^p|.$$

7.2 PUBLICATION OF STATIC DATASETS OF LOCATIONS

The work in [14] considers the publication of spatial datasets using two types of spatial index structures, namely, space-partitioning and data-partitioning indexes, respectively. Building an index structure can be seen as performing a series of split operations, and the result of these operations is a *spatial decomposition* of a dataset.

Depending on the type of spatial index used, each such operation can be determined by the data or not. For instance, as a representative of space-partitioning indexes, the quad-tree [16] is considered in [14]. Quad-trees are a popular structure for privacy solutions in location-based services; in fact in Chapter 2 we have seen two variations of cloaking techniques which rely on quad-trees: *IntervalCloak* and *Casper*. The main reason why quad-trees are used so frequently is their simplicity. In particular, in the case of differentially private spatial decompositions, the advantage of using quad-trees is that each split is performed independently on the actual location in the dataset. The data space is always split recursively on the middle coordinate of the current partition. Partitions are always square, and all the nodes in the tree at the same depth correspond to a dataspace partition of the same dimensions (and consequently, the same area or volume). Therefore, no information about the data is disclosed by the structure of the published tree structure, since it is completely independent of the data. According to the definition of differential privacy, since the structure will look identical regardless of which sibling dataset was used to create it, the outputs will be completely indistinguishable. All that is needed to privately publish a quad-tree indexed dataset according to the differential privacy model is to perturb each node count by adding Laplace noise.

In addition, quad-trees result in partitions that are non-overlapping. According to the sensitivity definition from Section 7.1, if one perceives each index partition, say a leaf node, as one of the elements that are part of the query set $\mathcal{Q}$, then all elements at the same level at the tree are non-overlapping, leading to constant sensitivity with low value 2. This is clearly a desirable property of such an index.

For the case of data-dependent index structures, the procedure required to obtain a differentially private version of a spatial dataset is more complex, because not only the individual counters

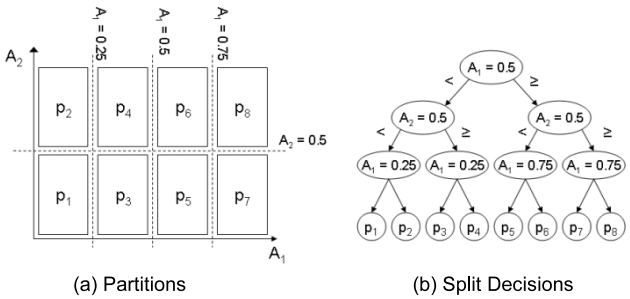
in each index node must be protected, but also the structure of the resulting tree. Different input data distributions may result to significantly different shapes of the index, which in turn can help an attacker infer that a certain dataset is more likely than others to be the actual dataset the index was built on. Additional measures such as protecting the result of each index split decision are necessary. We present next each of the two cases.

7.2.1 DATA-INDEPENDENT DECOMPOSITIONS: QUAD-TREES

In a d-dimensional space, quad-trees are constructed by recursively dividing the space into two sub-spaces using $(d-1)$-dimensional hyperplanes. For example, in 2D, each step corresponds to selecting a line and partitioning the space based on this line until certain requirements are met. In 3D, instead of lines, planes are used to split the space. The heuristic method for selecting the splitting hyperplanes depends on the application area, as does the stopping condition.

Each split occurs on an axis-orthogonal hyperplane. The axis is chosen based on the depth of the tree node which is being split. Specifically, if the depth of node n is denoted $Depth(n)$, the splitting hyperplane is orthogonal to axis $Depth(n)\%d + 1$ and splits the space into two equal halves.

Figure 7.1 exemplifies a partial quad-tree of height 3 in 2D space. The data space is first partitioned into equal partitions on A_1 axis through the line $A_1 = 0.5$. For all intermediate nodes at level 1, the next splitting line is orthogonal to A_2, $A_2 = 0.5$. At level 2, the splitting line partitions on A_1 axis again. The resulting partitions are the leaf nodes of the tree: $p_1, \ldots p_8$. Partitions cover the same volume of space and their regions do not depend on data distribution.



(a) Partitions (b) Split Decisions

Figure 7.1: Quad-tree example.

Let $\mathcal{Q}^h$ be a query that asks for the quad-tree index on some dataset D. Since partition extents are built based on attribute domains only, $\mathcal{Q}^h$ is equivalent to a set of 2^h count queries, one for each leaf node's extent. Notice that regardless of h and the distribution of D, the query regions will be disjoint. Therefore, the sensitivity of $\mathcal{Q}^h$ is 2 at each level of the tree.

In [14], the location count of each node in the quad-tree index of height h is published, regardless of whether it is a leaf or an internal node. Since each internal node covers a portion of

the dataspace that is exactly equal to the concatenation of partitions of its children, and there are no overlaps between nodes at the same level, it results that each point in the dataspace is covered by exactly h nodes, or equivalently, h queries. According to the sequential composition theorem of differential privacy [14], if every query at level i is answered with budget ϵ_i, then the publication of the entire tree satisfies differential privacy with parameter

$$\epsilon = \sum_{i=0}^{h} \epsilon_i \ ,$$

in other words the sum of all individual budgets at the tree levels.

One important decision to make remains how to split the budget across different levels of the tree. In other words, given a global privacy budget constraint ϵ, how to choose the values $\epsilon_i, i = 0 \ldots h$ to minimize the accuracy error when answering queries. Note that error is inevitable to achieve privacy, due to the addition of noise, but careful budget allocation can solve the problem in a satisfactory way.

In [14], the authors make the observation that given a random rectangular query Q over the dataspace domain, a closed-form, average-case scenario can be formulated for the number of index nodes $n(Q)$ that intersect query Q. Also, denote by $n_i(Q)$ the corresponding number of nodes at level i that maximally contribute their counts to the query result (i.e., their extents are completely enclosed by the query range Q). Therefore,

$$n(Q) = \sum_{i=0}^{h} n_i$$

It can be shown that, in the case of quad-trees, it holds that

$$n_i \leq 8 \cdot 2^{h-i}$$

and by summation over the entire height of the tree

$$n(Q) \leq 8(2^{h+1} - 1) = O(4^{h/2})$$

Solving a constraint optimization problem that minimizes the expected query answering error subject to the available maximum budget ϵ, the choice of geometric budget allocation over tree levels proves a good alternative; namely, the budget allocated to tree level i should be set to:

$$\epsilon_i = 2^{(h-i)/3} \epsilon \frac{2^{1/3} - 1}{2^{(h+1)/3} - 1}$$

7.2.2 DATA-DEPENDENT DECOMPOSITIONS: KD-TREES

As mentioned above, the additional challenge in the case of data-dependent indexing techniques is that the structure of the resulting index also discloses information about the dataset. Most of the

decomposition algorithms in literature involve a sequence of splits that is deterministic and depends on the actual data points. Since two relatively similar (possibly sibling) datasets may lead to significantly different outcome structures, an adversary may be able to reverse-engineer the deterministic process of index construction. Therefore, all decisions that are made in the process of index building must be differentially private as well.

In the case of kd-trees, the split algorithm recursively divides the space on the median value across one of the space dimensions. Instead of computing the actual median, a version of "noisy" median is determined using the exponential mechanism [41] of differential privacy. Specifically, each element in the dataset is assigned a merit score according to the distance it has from the actual median, and a randomized algorithm that picks an element according to these merit scores is used to select a noisy pseudo-median. Even though the pseudo-median may not be equal in value to the actual median, values closer to the actual median have higher chances of being selected.

As a consequence, another important change in comparison to the case of data-independent indexes is that the budget must be split between the Laplace mechanism and the exponential mechanism. An inherent trade-off develops: the more budget is allocated to the exponential mechanism, the index structure will be balanced, and improve accuracy. But on the other hand, there is less remaining budget to be used for releasing index node counts, which decreases accuracy.

For the budget allocated to releasing counts, a similar desirable budget allocation as for the case of quad-trees can be derived for kd-trees. According to [14], given the same expected average query Q, the number of expected kd-tree nodes covered by the query at level i is

$$n_i \leq 8 \cdot 2^{\lfloor (h-i+1)/2 \rfloor}$$

and by summation over the entire height of the tree

$$n(Q) \leq 8(2^{\lfloor (h-i+1)/2 \rfloor} - 1) = O(2^{h/2})$$

7.3 PUBLICATION OF TRAJECTORY DATASETS

As we discussed in Chapter 6, the problem of publishing trajectories is considerably more challenging than publishing location snapshots, due to the fact that the location sequencing and grouping into trajectories gives an adversary more vectors to successfully attack and break individual privacy. The same concept applies in the case of protecting trajectories with differential privacy as well, and differentially private publication of trajectories is more difficult than its location-snapshot privacy counterpart proposed in [14].

The problem of differentially private publication of trajectories was tackled only very recently in [10], where a case study is performed with respect to the feasibility of applying differential privacy to traces of real user trajectories from the public transportation database of the city of Montreal, Canada.

A data model where locations are discretized is considered, which models well a public transportation network, where each start, end, and intermediate snapshot in a user trajectory can be

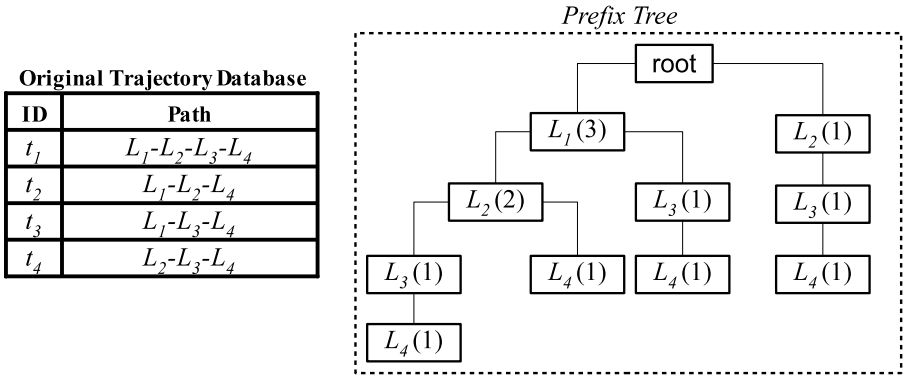


Figure 7.2: Trajectory database and resulting prefix tree.

identified with a bus or train stop. This is not a severe limitation to the model, since even in a more fine-grained model of movement, it is possible to first create a discrete space of logical locations (of arbitrary granularity), and then map continuous geographical coordinates to the discrete space. We follow the convention from [10], and denote a trajectory as

$$T = \{t_1, t_2, \ldots, t_{|T|}\}$$

where $|T|$ denotes the cardinality of trajectory T (i.e., the number of locations in the trajectory sequence), and all locations t_i are elements of a discrete set of locations $\mathcal{L}$, i.e., $t_i \in \mathcal{L}$.

A database of trajectories is a set of such objects T, one per each individual. The goal of differential private publication is to release a sanitized database of trajectories, such that given the output, an adversary is not able to tell if a particular individual's trajectory is included in the release or not.

The specific format of publication chosen in [10] is that of a prefix tree. Specifically, a tree is constructed, having a virtual root node corresponding to a special "start-of-trajectory" marker (which is not an actual location in $\mathcal{L}$). Starting from that node, a path in the tree is created for each individual trajectory. Trajectories that share an exact prefix will have identical beginning sub-paths in the tree.

Figure 7.2 shows an example of trajectory prefix tree, corresponding to the set of four trajectories in the table. The counter shown in parentheses for each node represents the number of trajectories that share that particular prefix (i.e., the prefix starting from the virtual root and ending in the respective node). Note that the prefix tree is constructed using a deterministic procedure from the dataset. Clearly, publishing this structure is not differentially private.

There are two types of disclosure that need to be prevented in order to obtain a data release that is compliant with differential privacy. First, the actual counters in the prefix tree need to be

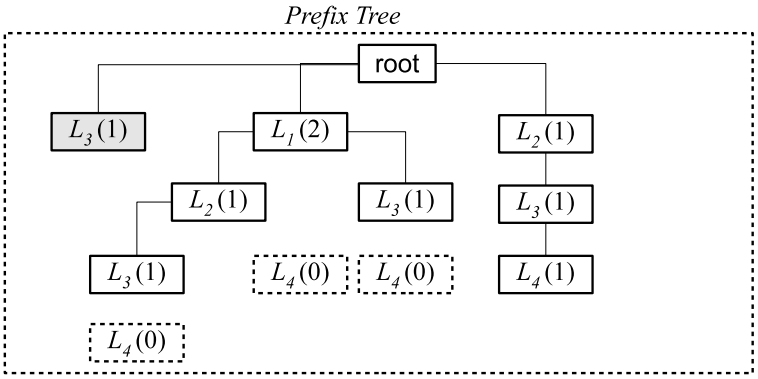


Figure 7.3: Differentially private prefix tree.

perturbed. This can be done using the Laplace mechanism, similar to the way that node counters were modified for quadtrees and kd-trees in [14]. Second, the actual structure of the prefix tree gives information about what particular trajectories exist and do not exist in the dataset. For instance, inspecting the tree in Figure 7.2, one can assert with certainty that there is no trajectory that starts at location L_3. Similarly, there is no trajectory that has as prefix L_1, L_4. To address this latter concern, it is necessary to alter the structure of the prefix tree and create fake nodes.

Figure 7.3 shows a possible tree generated after the sanitization step, where both counters and structure are perturbed. Note that, after sanitization, some nodes may have their count reduced to zero, so they are removed from publication, even though there are some trajectories in the original dataset represented by that path in the tree. Conversely, some nodes that do not correspond to any existing trajectory prefix are created. These fake nodes are necessary to comply with differential privacy.

In this example, the former leaf nodes L_4—all on the left half of the initial subtree—have had their counters added with negative noise -1, so as a result their noisy counter is zero, and they are removed from the tree (their perimeters are shown with discontinued line in the diagram for illustration purposes, but they do not appear at all in the published version). In contrast, for every possible child of a node—not only those in the initial prefix tree, but for all *possible* trajectories—a virtual node is created. This node has initial counter zero, but following addition of noise, they could be promoted to actual nodes. This is the case of node L_3, shown in shaded texture, which has been added as a child of the root node. Note however, that fake nodes can be added in any part of the tree, not just under the root node.

One particular case of structure protection occurs when making the decision to output the leaf nodes. Since fake nodes may be added and existing nodes may be deleted, the height of the tree can also vary from the original height. In order to prevent the adversary from inferring any

information from the tree height, it is suggested in [10] to limit the maximum trajectory length, and thus arbitrarily limit the tree height, to a constant, a system parameter which is *independent* of the dataset, and thus safe to disclose.

Clearly, adding and removing nodes, and also modifying the counters in the initial nodes, introduces error in the representation of the dataset. This error is necessary for privacy protection. Experimental results from [10] show that the error can be high, particularly as the maximum allowed trajectory length is large. Nevertheless, the technical approach of using a prefix tree is an interesting first step in differentially private publication of trajectory data, and future research is likely to improve on the obtained error through optimizations.

CHAPTER 8

Conclusions

Location privacy has already been widely acknowledged as an important problem, both by the research community, as well as by various legislative bodies. Governments in the European Union, United States, and Asia-Pacific have all started some form of initiative to mandate privacy requirements for service providers that handle individuals' location data. Awareness on the issue of location privacy is already heightened, and it is certain that in the future effective privacy-preserving solutions will be necessary to support the widespread development and adoption of location-based applications. Although neither a technical, nor a policy-based universal approach exists to deal with this difficult problem, many efforts in the research community have shown encouraging results at least in solving some of the sub-problems, and addressing certain application scenarios. These lecture notes have surveyed some of the most prominent solutions, and illustrated several important facets of location privacy, ranging from protection for location-based queries, to private spatial matching, trajectory publication, and differentially private approaches.

For the private location-based queries domain, we have identified a taxonomy of query protection techniques and highlighted their relative privacy-performance trade-offs. At one end of the spectrum, two-tier spatial transformation methods such as SpaceTwist and dummy-generation schemes incur low overhead, but they only provide privacy under a limited set of assumptions (i.e., they only provide protection against attackers with no background knowledge). At the opposite end, the cryptographic PIR approach offers strong privacy even in highly adversarial scenarios, both for snapshot and continuous queries, but its overhead can be quite significant, especially for large datasets. Finally, the spatial anonymity model adopted by three-tier spatial transformation techniques manages to provide a good amount of privacy for snapshot queries, and incur moderate overhead.

For the future of private location-based queries, hybrid approaches seem to be the best direction to follow, because they provide strong cryptographic guarantees as long as the user moves within a certain range of the dataspace. With a properly chosen range, it can be ensured that no information about an individual's whereabouts is released that can be used to infer sensitive information, and at the same time the overhead of expensive cryptographic operations can be held in check. As a third dimension of the required properties of query protection methods, hybrid approaches also bring the benefit of protecting excessive disclosure of data from the provider, which not only protects the privacy of the users, but also the commercial interest of the service providers.

With the emergence of an increasingly larger number of online geosocial networks, and more broadly applications that generate large amounts of geo-tagged social media, the issue of private matching of spatial datasets gains importance as well. We have seen how certain carefully chosen geometric transformations can be used to match spatial datasets using a proximity-metric

function with good accuracy and little disclosure. However, more complex join conditions may be required, and therefore more sophisticated techniques will be needed in the future, with increased flexibility in the matching condition, as well as with superior, and hopefully analytically bounded data disclosure guarantees. One interesting direction to follow is to bring together techniques from data mapping and feature selection with secure multi-party computation (SMC) algorithms. While SMC algorithms are notoriously expensive to be used on their own, creating a hybrid approach where alternate methods can be used to filter out most of the input datasets before applying SMC seems to be a promising direction to follow.

The research area of privacy-preserving trajectory publication is currently fractured between two different paradigms: releasing independent location samples versus publishing entire trajectories. The advocates of the former approach claim that generalization and suppression of data, which is inherent if individual trajectories are published, cannot achieve the level of accuracy dictated by practical applications. On the other hand, publication of independent samples may not be sufficient if complex tasks (e.g., deriving travel patterns) are performed on top of the data.

Generalization-based mechanisms for trajectory publication are also faced with the challenging problem of protecting against attackers with background knowledge, and this is the main reason why significant future results will most likely not come from the family of generalization methods, due to the inherent limitations of syntactic privacy models. On the other hand, the direction of location and trajectory publication in the setting of differential privacy, although at an early stage currently, is a more appealing approach, due to its robustness and the formal semantic privacy model that deals well with background knowledge attacks. Most likely, future research efforts in the direction of location dataset publications should focus on differential privacy, and the numerous customization and optimization approaches that have been recently proposed for relational data (e.g., medical records) could be adapted to improve the accuracy of differentially private releases of geospatial datasets as well.

Bibliography

[1] *The Octopus Payment System:* `http://www.octopuscards.com`. 1

[2] O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *ICDE*, pages 376–385, 2008. DOI: 10.1109/ICDE.2008.4497446 52, 53

[3] N. R. Adam and J. C. Wortmann. Security-Control Methods for Statistical Databases: A Comparative Study. *ACM Computing Surveys*, 21(4):515–556, 1989. DOI: 10.1145/76894.768959

[4] C. C. Aggarwal. On k-Anonymity and the Curse of Dimensionality. In *Proc. of VLDB*, pages 901–909, 2005. 10

[5] R. Agrawal and R. Srikant. Privacy-Preserving Data Mining. In *SIGMOD*, 2000. DOI: 10.1145/335191.335438 9

[6] R. Bayardo and R. Agrawal. Data Privacy through Optimal k-Anonymization. In *Proc. of ICDE*, pages 217–228, 2005. DOI: 10.1109/ICDE.2005.42 10

[7] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003. DOI: 10.1109/MPRV.2003.1186725 10

[8] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 2004. DOI: 10.1109/PERCOMW.2004.1276918 47

[9] A. R. Butz. Alternative Algorithm for Hilbert's Space-Filling Curve. *IEEE Trans. Comput.*, C-20(4):424–426, 1971. DOI: 10.1109/T-C.1971.223258 6

[10] R. Chen, B. C. Fung, B. C. Desai, and N. M. Sossou. Differentially private transit data publication: a case study on the montreal transportation system. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 213–221, 2012. DOI: 10.1145/2339530.2339564 55, 61, 62, 64

[11] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar. Preserving User Location Privacy in Mobile Data Management Infrastructures. In *Proc. of Privacy Enhancing Technologies*, pages 393–412, 2006. DOI: 10.1007/11957454_23 11

[12] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995. 15

[13] C.-Y. Chow, M. F. Mokbel, and X. Liu. A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Services. In *Proc. of ACM International Symposium on Advances in Geographic Information Systems*, 2006. DOI: 10.1145/1183471.1183500 12, 27

[14] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 20–31, 2012. DOI: 10.1109/ICDE.2012.16 55, 58, 59, 60, 61, 63

[15] M. Damiani, E. Bertino, and C. Silvestri. The PROBE Framework for the Personalized Cloaking of Private Locations. *Transactions on Data Privacy*, 3(2):123–148, 2010. 5, 7, 26, 27

[16] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2nd edition, 2000. 19, 20, 31, 55, 58

[17] C. Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006. 55, 56

[18] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006. DOI: 10.1007/11681878_14 56, 57

[19] D. E. Flath. *Introduction to Number Theory*. John Wiley & Sons, 1988. 16, 17

[20] B. Gedik and L. Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *Proc. of ICDCS*, pages 620–629, 2005. DOI: 10.1109/ICDCS.2005.48 5, 10, 11, 26, 27

[21] G. Ghinita, M. L. Damiani, C. Silvestri, and E. Bertino. Preventing velocity-based linkage Attacks in location-aware applications, *International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS)*, 2009. xi, 7

[22] G. Ghinita, P. Kalnis, M. Kantarcioglu, and E. Bertino. A Hybrid Technique for Private Location-Based Queries with Database Protection, *SSTD*, 2009. xi, 25, 26

[23] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. L. Tan. Private Queries in Location Based Services: Anonymizers are not Necessary. In *SIGMOD*, 2008. DOI: 10.1145/1376616.1376631 xi, 5, 15, 22, 23, 25, 27, 31, 34

[24] G. Ghinita, P. Kalnis, and S. Skiadopoulos. MobiHide: A Mobile Peer-to-peer System for Anonymous Location-based Queries. In *SSTD*, 2007. DOI: 10.1007/978-3-540-73540-3_13 27

[25] G. Ghinita, P. Kalnis, and S. Skiadopoulos. PRIVE: Anonymous Location-based Queries in Distributed Mobile Systems. In *WWW*, 2007. DOI: 10.1145/1242572.1242623 27

[26] G. Ghinita, C. R. Vicente, N. Shang, and E. Bertino. Privacy-preserving matching of spatial datasets with protection against background knowledge. In *International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS)*, 2010. DOI: 10.1145/1869790.1869795 xi, 36

[27] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. of USENIX MobiSys*, 2003. DOI: 10.1145/1066116.1189037 5, 10, 26, 27

[28] M. Gruteser and X. Liu. Protecting Privacy in Continuous Location-Tracking Applications. *IEEE Security and Privacy*, 2:28–34, 2004. DOI: 10.1109/MSECP.2004.1281242 2, 26, 27, 47

[29] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 2005. 51

[30] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *Proc. of International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM)*, pages 194–205, 2005. DOI: 10.1109/SECURECOMM.2005.33 12, 48

[31] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proc. of the 6th international conference on Mobile systems, applications, and services (MobiSys)*, pages 15–28, 2008. DOI: 10.1145/1378600.1378604 50

[32] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in gps traces via uncertainty-aware path cloaking. In *Proc. of the 14th ACM conference on Computer and Communications Security (CCS)*, pages 161–171, 2007. DOI: 10.1145/1315245.1315266 49, 50

[33] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preserving Location-based Identity Inference in Anonymous Spatial Queries. *IEEE TKDE*, 19(12), 2007. DOI: 10.1109/TKDE.2007.190662 xi, 5, 12, 18, 26, 27

[34] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing Source-Location Privacy in Sensor Network Routing. In *Proc. of ICDCS*, pages 599–608, 2005. DOI: 10.1109/ICDCS.2005.31 12

[35] A. Khoshgozaran and C. Shahabi. Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy. In *SSTD*, 2007. DOI: 10.1007/978-3-540-73540-3_14 5, 6

[36] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *International Conference on Pervasive Services (ICPS)*, pages 88–97, 2005. DOI: 10.1109/PERSER.2005.1506394 5, 6

[37] E. Kushilevitz and R. Ostrovsky. Replication is NOT needed: Single database, computationally-private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 364–373, 1997. DOI: 10.1109/SFCS.1997.646125 15, 17

[38] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient Full-Domain K-Anonymity. In *SIGMOD*, 2005. DOI: 10.1145/1066157.1066164 10

[39] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proc. of ICDE 2007*, pages 106–115, 2007. DOI: 10.1109/ICDE.2007.367856 55

[40] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *ICDE*, 2006. DOI: 10.1109/ICDE.2006.1 7, 10, 55

[41] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, pages 94–103, 2007. DOI: 10.1109/FOCS.2007.41 61

[42] A. Meyerson and R. Williams. On the Complexity of Optimal K-anonymity. In *Proc. of ACM PODS*, pages 223–228, 2004. DOI: 10.1145/1055558.1055591 12

[43] M. F. Mokbel, W. G. Aref, and I. Kamel. Analysis of Multi-Dimensional Space-Filling Curves. *GeoInformatica*, 7(3):179–209, 2003. DOI: 10.1023/A:1025196714293 12

[44] M. F. Mokbel, C. Y. Chow, and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *Proc. of VLDB*, 2006. 5, 10, 26, 27

[45] B. Moon, H. Jagadish, and C. Faloutsos. Analysis of the Clustering Properties of the Hilbert Space-Filling Curve. *IEEE TKDE*, 13(1):124–141, 2001. DOI: 10.1109/69.908985 12

[46] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999. DOI: 10.1007/3-540-48910-X_16 27, 32

[47] D. Reid. An Algorithm for Tracking Multiple Targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979. DOI: 10.1109/TAC.1979.1102177 47, 48

[48] P. Samarati. Protecting Respondents' Identities in Microdata Release. *IEEE TKDE*, 13(6):1010–1027, 2001. DOI: 10.1109/69.971193 5, 9, 51

[49] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990. 10

[50] Y. Saygin, E. Nergiz, and M. Atzori. Towards trajectory anonymization: a generalization-based approach. In *International Workshop on Security and Privacy in GIS and LBS (SPRINGL)*, pages 52–61, 2008. DOI: 10.1145/1503402.1503413 53

[51] L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002. DOI: 10.1142/S0218488502001648 5, 9, 51, 55

[52] Y. Tao and D. Papadias. Historical spatio-temporal aggregation. *ACM Trans. Inf. Syst.*, 23(1):61–102, 2005. DOI: 10.1145/1055709.1055713 13

[53] Y. Tao and X. Xiao. Personalized Privacy Preservation. In *Proc. of ACM SIGMOD*, 2006. DOI: 10.1145/1142473.1142500 10

[54] M. Terrovitis and N. Mamoulis. Privacy preservation in the publication of trajectories. In *Proc. of International Conference on Mobile Data Management(MDM)*, pages 65–72, 2008. DOI: 10.1109/MDM.2008.29 50, 51, 52

[55] X. Xiao and Y. Tao. Output perturbation with query relaxation. *PVLDB*, 1(1):857–869, 2008. 57

[56] M. L. Yiu, C. Jensen, X. Huang, and H. Lu. SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. In *International Conference on Data Engineering (ICDE)*, pages 366–375, 2008. DOI: 10.1109/ICDE.2008.4497445 5, 6

Author's Biography

GABRIEL GHINITA

Dr. Gabriel Ghinita is an Assistant Professor with the Department of Computer Science at University of Massachusetts, Boston (UMB). Prior to joining UMB, he was a Research Associate affiliated with the Computer Science Department at Purdue University, and the Purdue Cyber Center. He holds a Ph.D. degree in Computer Science (2008) from the National University of Singapore, and a BS degree in Computer Science (2003) from "Politehnica" University of Bucharest.

Dr. Ghinita's research interests lie in the area of databases, with focus on information security and privacy. His work includes research papers on Anonymous Publication of Geospatial, Relational and Set-valued data, Privacy-Preserving Sharing of Location Data, Secure Data Outsourcing and Secure Data Provenance, and Trustworthiness Assessment. He is also interested in spatio-temporal databases and data management in large-scale distributed environments.

Dr. Ghinita's professional service includes participation on the Program Committee of top database conferences (ACM SIGMOD, PVLDB, ICDE), as well as reviewing for journals such as VLDBJ, IEEE TKDE, IEEE TPDS, IEEE TMC, IEEE TDSC, and GeoInformatica.