Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Michael G. Solomon Date

Efficient Search and Computation on Encrypted Data with Access Control

By

Michael G. Solomon
Doctor of Philosophy

Computer Science and Informatics

Vaidy Sunderam, Ph.D.
Advisor

Li Xiong, Ph.D.
Co-advisor

James Lu, Ph.D.
Committee Member

Ming Li, Ph.D.
Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the Graduate School

Date

Efficient Search and Computation on Encrypted Data with Access Control

By

Michael G. Solomon
Ph.D., Emory University, 2016

Advisor: Vaidy Sunderam, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2016

Abstract

Efficient Search and Computation on Encrypted Data with Access Control
By Michael G. Solomon

Outsourcing data and processing to cloud environments often raises security and privacy concerns, which can be addressed through the use of encryption. But current approaches either provide all-or-nothing encryption, or rely on an omniscient third party to handle granular key management and make access control decisions to provide fine-grained access control, and introduce obstacles to searching over ciphertext. We explore the problem of efficiently searching encrypted data and simultaneously providing embedded fine-grained access control, first in a general setting, and then extended to location-based data. We first propose a new framework for generic database data that enforces access control for queries from different classifications of users, while still providing the capability to search over encrypted data. We then extend our research focus to location-based applications by implementing and assessing several existing location privacy solutions to produce concrete recommendations of the best technique for implementors to choose for specific use cases. And finally, we combine the first and second parts of our work to propose another new framework for mutually private proximity detection (MPPD) to efficiently support searching over encrypted data and enforcing fine-grained access control and privacy for data owners (DO) and users for location-based applications. The culmination of our work provides researchers and application developers with a viable framework that provides MPPD in a categorical setting, and is based on current architectures and technologies.

Efficient Search and Computation on Encrypted Data with Access Control

By

Michael G. Solomon
Ph.D., Emory University, 2016

Advisor: Vaidy Sunderam, Ph.D.

A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2016

Acknowledgments

I would like to thank my co-advisors Dr. Vaidy Sunderam and Dr. Li Xiong, for supporting my research and challenging me throughout this journey. They both provided me with guidance and inspiration, and taught me more than I can ever express. I would also like to thank my committee members, Dr. James Lu, and Dr. Ming Li, for their advice and challenges that helped to focus and refine my research. I could not have reached this goal without input and support from each one. I would also like to thank all of my fellow Math & CS students for their friendship, encouragement, and inspiration. In many cases, it was their examples that provided the motivation to dig in and move ahead. And I could not have navigated through the myriad requirements without the unwavering support and help from Terry Ingram, Erin Nagle, and all of the Math & CS department staff. And finally, I want to extend my deepest gratitude to my most fervent cheerleaders, Stacey, Noah, and Isaac, who always have shared my pursuit of this goal. I am thankful for all who have helped me along the way.

In memory of Isaac Samuel Solomon 1993-2014.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The rapid growth of data generated and consumed by diverse applications and users has led to large scale data and processing outsourcing. Many of today's applications execute, and access data stored in, cloud environments, hosted by cloud service providers (CSP) [31]. An agreement with a CSP to store data in a public, community, or hybrid cloud environment can provide the benefits of outsourced maintenance and capability to alter capacity based on demand [12]. However, diminished control over data security and privacy [61, 37] is a disadvantage of outsourcing data storage and processing. CSP environments are untrusted [24] in which local levels of control cannot be attained [38, 37]. Traditional access control methods are often insufficient for CSP [38, 45] hosted databases and processing. Lacking sufficient confidentiality and privacy controls not only exposes the data to additional vulnerabilities, but is also potentially a violation of laws, regulations, or contract terms [54]. The primary challenges are to provide increased and more granular control over who can consume data, while relocating data and processing outside traditional trust boundaries.

The first part of this dissertation addresses the challenge of providing secure fine-grained access control in categorical settings, that is, in settings that

support groups of users with different access needs. Users are commonly authorized to access data based on characteristics or membership in certain groups. A common method to protect data in any untrusted environment is to encrypt data before sending it outside the trusted domain [22]. In multi-user database scenarios, solutions using most traditional encryption implementations are suboptimal, requiring an additional key-management layer thereby degrading performance and scalability [79, 49, 20]. For example, if two users, Alice and Bob, who both possess the attributes of belonging to an authorized group, want to share data stored in a cloud database using traditional encryption methods, they must both have access to the single key necessary to decrypt data, and their access to that key must be controlled by some third party. This approach places the access control decisions on the entity that controls access to decryption keys. We address this problem by proposing a framework that provides data confidentiality and categorical access to data without incurring the overhead of traditional complex key management.

In addition to the need to control general access to data based on user characteristics, applications deployed on mobile devices increasingly incorporate location awareness into services they provide. In fact, many of these services rely on location to provide context-sensitive results, such as proximity alerts when a consumer approaches some defined area of interest (AOI). Such applications can provide guidance to users based on the context of their current real world environment. AOIs can define locations that users want to approach, such as a store with an attractive sale, or places to avoid, such as a traffic accident or other event that threatens public safety. As smart phones, tablets, and other mobile devices approach ubiquity, detecting and tracking device location and providing location-specific services becomes increasingly easier. Different location detection techniques [47, 60] provide various levels of accuracy in different environments (indoor/outdoor, above/below ground,

urban/rural, etc.). Today's smart phones, tablets, and other mobile devices make accurate location-sensing a commonly used feature. However, consumers of such features are becoming increasingly concerned over the loss of privacy resulting from ongoing location disclosure. In short, consumers want to enjoy the benefits of location-based services without disclosing their location. Likewise, data owners may want to provide proximity notification without publishing the locations of AOIs, to protect intellectual property or preserve public safety. Mobile apps often include location based services, such as proximity to preferred vendors or notification of safety issues, but require users to disclose their locations. Users want to know "am I near an area of interest? (e.g. certain type of restaurant or a bank)", and might also desire push notifications when they are near areas of interest, areas to avoid for safety reasons, or any area with associated valuable location specific information. Users value such services but still want to keep their locations private. Location exposure privacy issues are being increasingly recognized including tracking, identification, and profiling threats [25, 14] . In the second part of this dissertation we evaluate several proposed solutions to providing mutually private proximity detection (MPPD). Our evaluation includes implementing a subset of the approaches evaluated, and then measuring the privacy levels and performance of each implemented approach as input data domains were varied, to develop a recommendation of the best fit approach for specific deployment environments.

As data owners create larger sets of AOIs for increasingly diverse groups of consumers, it becomes more important to provide differing levels of service based on consumer classifications, without disclosing the locations of all areas of interest to all consumers. For example, data owners may provide services of greater value to paid subscribers than to users who consume services for free. Data owners want to restrict access to, and even awareness of, areas of interest to consumers based on access permissions and location.

In the third part of this dissertation we explore the next logical step, that is to combine MPPD approaches with categorical access to searchable data. This third and final part of our research proposes a framework that meets the usability requirements of today's location-based services, while providing mutual privacy for users and data owners, with embedded access rules based on data owner defined access policies.

1.1.1 Confidentiality in Categorical Settings

A primary challenge when outsourcing data or processing to cloud environments is to extend both confidentiality assurances and control of privacy into untrusted domains [45]. Since different data consumers likely have different privileges, data access must be individualized and restricted only to authorized consumers. And to be functionally effective, the protected data must be searchable without incurring excessive overhead or exposing any of the protected data to any entities in the untrusted environment [17].

Traditional data encryption techniques require a single key, or a pair of keys, to encrypt and decrypt each data item. The most fine-grained approach to using encryption for data stored in a database requires a separate key for each cell (each column within a row), and a trusted key authority to store keys and manage access to them based on pre-determined access criteria. A more common practice is to reuse keys for groups of cells, reducing the total number of keys and required administrative overhead to manage them. Even this more realistic case would require a trusted omniscient third party key access manager to store keys and manage access to them based on some pre-determined access criteria. The added overhead of managing key associations and access rights for many users makes it difficult to scale. The opposite extreme approach would be to use a single key or key pair to encrypt and decrypt all protected cells in the database, an approach similar to various

transparent data encryption (TDE) schemes [27, 19]. This approach makes it easier to manage keys but introduces a single point of compromise. Any user who possesses the single key can submit and consume all data stored in the database. A balance between the two extremes is to define partitions of encrypted data (the set of encrypted cells in a database that share the same encryption/decryption key), and are often implemented as roles [77]. Suppose a user, U_2 as shown in Figure 1.1, owns the key for partition P_2, (K_2), and also wants to access data in partition P_1. User U_2 would request the key for P_1, (K_1), from the key access manager (or directly from user U_1) and then use it to decrypt data from the database. If user U_2 only possessed the keys for partitions P_1 and P_2, (K_1, K_2), she could not access any encrypted data from any other partition. Associating partition keys with users provides the ability to expose subsets of the database to users, while maintaining confidentiality of partitions any user is not authorized to access.

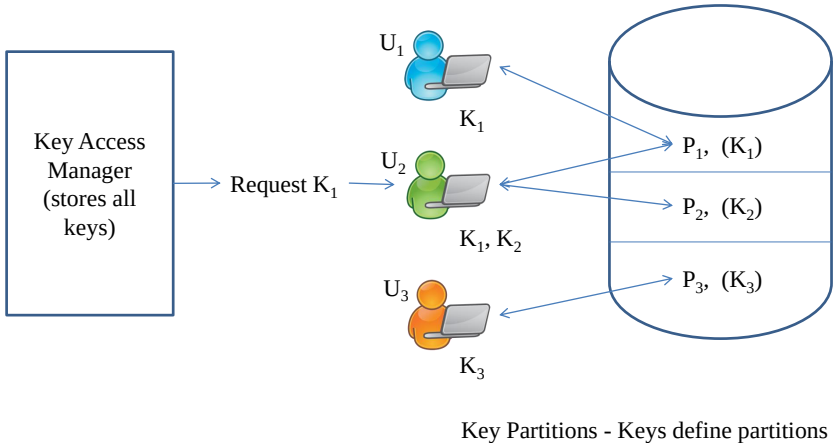


Figure 1.1: Accessing Data From Multiple Partitions

While this approach is a good compromise between minimum and maximum granularity, the common use of key access managers does still grant access control authority to the key access manager, instead of giving the authority

to the data owner.

1.1.2 Location Privacy

Consider a frequent traveler, Alice, with a GPS enabled smartphone. Alice wants to know when she is near one or more areas of interest (AOIs), without disclosing her location. AOIs to avoid may include areas of traffic accidents, infectious diseases, civil emergencies, or even weather events. The locations of weather AOIs are public and do not need privacy protection. However, authorities may want to keep the locations of some other types of AOIs private, such as areas of known criminal activity, to protect both the public and law enforcement officers, or ongoing investigation, to help investigators maintain the integrity of a crime scene. The data owner (i.e. law enforcement agency) would likely maintain separate sets of AOIs, one for public consumption and another for internal personnel consumption. The AOI set for the general public would likely contain larger AOIs to provide more general information, and the AOI set for internal personnel would contain smaller AOIs to convey more precise location information. The law enforcement agency wants to protect their AOI locations until a user has a "need to know" when approaching an AOI. Users would satisfy this property when they are in proximity to an AOI defined in an AOI set for which they are authorized. We refer to this problem as MPPD, which detects proximity of a user to defined AOIs without disclosing the user's location to the data owner or any AOI information to the user. We note that even with MPPD, any user (or multiple colluding users) will learn some AOI location information *after* receiving a proximity alert and potentially construct an AOI map as they move around in the space. As an orthogonal issue, AOI privacy depends on either the AOI locations changing before users can infer material portions of the AOI map or the use of compensating controls. For example, law enforcement may

enforce physical boundaries around an investigation to prevent unauthorized individuals from learning anything more than the fact that "something is going on" in a general area.

1.1.3 Location Privacy in Categorical Settings

The previous section described a need for MPPD in applications that provide proximity detection for all users within a defined set of AOIs. But as location-based applications and devices become more commonplace, it becomes more important to provide MPPD along with controlled access to AOIs. In other words, it is desirable to allow data owners to define AOIs and specify access restrictions based on user characteristics. Consider Mary, who is a guest at the "Fun Times" amusement park. Mary wants to make the most of her day in the park and desires to minimize time spent waiting in line for attractions or shows. Mary has subscribed to the "Fun Times InTheKnow app" premium service that sends information to her smart phone about nearby attractions and shows with short wait times. Bob is also in the "Fun Times" park and has the "InTheKnow" app, but Bob did not subscribe to the premium service. Bob only receives general information about attraction wait times for attractions that are somewhat close to his current location. Both Mary and Bob want to receive helpful information but are concerned about openly broadcasting their every move within the park to the park's management. On the other hand, "Fun Times" wants to provide location-sensitive services to Mary and Bob without publishing all of the "Fun Times" areas of interest. "Fun Times" can limit the information they provide to subscribers, and even define different subscriber levels based on subscription fees paid. "Fun Times" also uses this service to direct their employees to areas of the park that need cleaning, servicing, or even crowd management. Other use cases for such a MPPD framework in a categorical setting could include providing

epidemiologic related alerts or criminal activity investigation proximity as described in the previous section, but with precision granularity based on clearance and/or "need to know".

1.2 Contributions

1.2.1 Cloud Database Confidentiality with Fine-grained Access Control (Chapter 3)

To meet the needs of protecting the confidentiality of outsourced data when being accessed by various categories of users, based on user attributes, we propose a framework that provides confidentiality in an untrusted environment along with maintaining *data owner control* over *data consumer (user) access* without an omniscient key manager. Our framework, termed ZeroVis[1] [62], combines the ability to search across encrypted data [58] with fine-grained access control [4] to provide confidentiality protection, searchability for efficient access, and data owner initiated access control, all in an untrusted storage environment. Our framework provides a one-to-many (one data owner to many data consumers) data confidentiality layer that can be accessed by existing legacy applications to allow current host-bound applications to migrate to a cloud storage environment and maintain confidentiality.

Our framework does not require a trusted third party to manage encryption keys for data owners and consumers. Nor does it require specific permission for each new data consumer (e.g. research team). In essence, each data owner (e.g. patient) specifies an access policy (based on attributes rather than identities) for her data that determines who can access protected portions of her data. Traditional key management schemes require a key manager

[1]Like flying an instrument approach with limited or no visibility - only pilots with proper equipment, clearance, and the current local frequencies can land.

to associate authorized data owners and authorized data consumers with keys (a many-to-many relationship). Our framework assumes the existence of an attribute manager that maintains valid attributes for authorized data consumers (instead of many keys), regardless how many partitions they can access.

Our framework proposes the use of CP-ABE (Ciphertext Policy Attribute Based Encryption) [4] to control access to data based on the data consumer's attributes. Only consumers who possess attributes that satisfy the ciphertext's access policy can decrypt. Our framework also utilizes layered encryption in combination with CP-ABE to support efficient query processing on encrypted data. We present an implementation of our framework and a performance study with different database sizes that demonstrate the feasibility of our proposed approach.

1.2.2 Mutually Private Proximity Detection (Chapter 4)

Naive solutions to protect location privacy include simply stopping location sensing or the release of physical location. However, this results in the loss of all utility (e.g. navigation or lost device tracking). An alternative approach is to introduce novel ways of exchanging and processing location information that protects the location owner's privacy without required end user action. Researchers have focused efforts to provide location privacy at the application layer in four main areas, each with its own drawbacks: location perturbation (limited accuracy), access control (limited privacy/utility trade-off), private information retrieval (PIR) (lack of data owner privacy), and encryption (performance cost). Each approach attempts to provide the ability for users to consume location based services without disclosing their locations to a service provider (SP) or data owner, and in some cases, allow the data owner

to keep its location data private as well. There is a lack of independent and objective comparative assessment of these techniques in the literature, in terms of their privacy, security, and performance. Our goal in this second part of the dissertation is to provide such an analysis and assessment of a subset of the techniques, with a view to evaluating their functionality and performance of a subset of techniques.

In our assessments [63] we focus on techniques guaranteeing accuracy as well as mutual privacy, i.e. those that are encryption based, and objectively evaluate their known deficiency, i.e. cost. In selecting the approaches to evaluate, we chose the algorithms from the "compute-then-compare" algorithm class. Other algorithms, such as Wang's circular range search [69], do not require separate computation and comparison steps. The relative proximity feature is embedded in the encoding method and is beyond the scope of this dissertation. We choose five representative techniques, and implement three of the selected techniques and evaluate how each implemented technique performs with varying input data (AOI number and size, and area coverage.) Organizations implementing a MPPD solution can use our results to select the best technique to fit how their environment changes over time. We present our results in a manner that provides clear direction for selecting the "best" technique for a specific application environment. As part of the algorithm identification process, we found that one approach held promise, but was not designed specifically to address physical location proximity queries. It was originally proposed as a general Secure k-Nearest Neighbor technique. The original technique determined closeness between vectors of attributes. We extended the technique's design to support distinct locations and range queries, while still preserving the mutual privacy of the user and the data owner. Our modifications adapted the technique to our problem domain and resulted in a new viable solution.

1.2.3 Mutually Private Proximity Detection in Categorical Settings (Chapter 5)

To address the problem of MPPD in categorical settings, we propose a framework and protocol, PrivProxABE [63], that allows data owners to define a single set of areas of interest (AOI), along with consumer access rules for each AOI. Embedded AOI access rules provide the ability to limit AOI access to specific groups of consumers. The locations of all defined AOIs are kept private and are not disclosed to service consumers. The framework also supports proximity detection based on a consumer's current location, while keeping the consumer's location private. That is, a consumer can issue a proximity query without disclosing current location, and still receive a result that indicates whether the consumer is near any area of interest for which the consumer is authorized. Our framework supports mutual privacy (privacy for the data owner and the consumer), along with embedded access control that allows data owners to control proximity alerts by consumer type.

Our protocol uses CP-ABE to provide fine-grained access control based on descriptive consumer attributes, and Hidden Vector Encryption (HVE) to efficiently determine user proximity to AOIs. To the best of our knowledge, there is only one other proposed protocol [51] that controls access to AOIs based on access rules the data owner supplies, along with MPPD. Li's Paillier based protocol requires distance calculations for each AOI. Our protocol reduces computational overhead by using HVE with compressed AOI location tokens to determine if a user's location overlaps one or more AOIs, along with CP-ABE to provide flexible fine-grained access control.

Our proposed protocol is novel in that it provides fine-grained flexible consumer access control, minimizes computational load on devices with limited processing power, (e.g. mobile devices), and also provides a high level of privacy guarantees for both users and data owners. The protocol supports these

services and reduces the data owner administrative workload. Data owners can create a single set of AOIs, along with access policies for each AOI, that restricts user consumption of proximity alerts based on data owner defined access policies. Users (consumers) must possess the attributes necessary to satisfy a data owner defined access policy for each AOI to receive proximity alerts for that AOI. All of this functionality is provided without requiring any third party to access unencrypted location information to make access authorization decisions. Using CP-ABE, data owners can define the granularity of user proximity alert consumption based on its own defined access policies. The use of HVE, along with limiting any public key distribution to only trusted entities, provides the actual proximity determination of a user location to an AOI without disclosing either location to any party. The combination of CP-ABE and HVE provides a flexible and scalable approach to implementing MPPD in a categorical user setting.

1.3 Organization

The remainder of the dissertation is organized as follows. In Chapter 2, we discuss the most related works to the problems addressed in this dissertation. In Chapter 3, we describe our contributions in preserving confidentiality in categorical settings for data stored in cloud databases. In Chapter 4 we examine the performance of three implementations of approaches that provide mutually private proximity detection. We conclude our contribution to private proximity detection in Chapter 5 where we present our mutually private proximity detection framework for categorical settings. In Chapter 6, we finally summarize the contributions of this dissertation and propose future research directions.

Chapter 2

Related Work

In this chapter, we summarize the related work most relevant to the contributions presented in this dissertation.

2.1 Encryption Key Management

The ZeroVis and PrivProxABE frameworks proposed in this dissertation both minimize overhead associated with distributed/federated encryption key management. Both frameworks provide fine-grained access control, with ZeroVis focused on general cloud database data and PrivProxABE focused on location-based shared data.

2.1.1 Sharing Encryption Keys

Researchers have addressed the problem of controlling access to encrypted data using a variety of methods. Early solutions were based on various types of encryption key sharing schemes. Broadcast encryption [28] was first proposed as a solution to the problem of sending secure transmissions from one site to an arbitrary number of recipients. This scheme relies on a known hierarchical key distribution pattern and set of privileged users. Later work based on [28] increases scalability [57] [44] and even integrates key derivation techniques for greater utility [78]. Since a requirement for our frameworks

is to minimize overhead associated with traditional key management, we do not incorporate key sharing in our proposed solutions.

2.1.2 Deriving Encryption Keys

Another approach to controlling access to encrypted data is to derive keys, as opposed to broadcast encryption techniques that manage key sharing. An early attempt at managed key derivation was proposed as Identity Based Encryption (IBE) [7]. IBE is a public key encryption technique in which keys are derived based on a user's identity. IBE removes the requirement that encryption keys must be securely shared with the intended recipient, but still assumes a single recipient. Goyal [32] proposed an extension to IBE, called Attribute Based Encryption (ABE), that uses attributes and access policies, not distinct identities, to encrypt and decrypt data. ABE addressed the problem of encrypting data for an arbitrary number of recipients. The two primary forms of ABE are Ciphertext Policy ABE (CP-ABE) and Key Policy ABE (KP-ABE). The difference between the two approaches is in how the access policy is used. There are 2 main ABE alternatives, defined by the location of the access policy. KP-ABE embeds the access policy in the user's private key [32], while CP-ABE embeds the access policy in the ciphertext [4]. KP-ABE gives control over who can decrypt data to the key generator, while CP-ABE ensures that the encryptor (data owner) retains control over who can decrypt her data [4]. ABE solves the problem of providing access to private data for a specified recipient without traditional key management issues, and is proposed in several outsourcing secure data schemes [74, 36, 75], but the technique alone does not map well to encrypting data for storage in a database due to its lack of a mechanism to efficiently search encrypted data. Li et. al. [50] uses both CP-ABE and KP-ABE schemes to store personal health record (PHR) data in a semi-trusted environment. Their proposed

framework extends the basic ABE notion to include Multi-Authority ABE (MA-ABE) [13] to allow different attribute authorities with different data needs to collectively generate users' secret keys based on distinct sets of user attributes. This approach of securing PHR data focuses primarily on storing documents and does not address the problem of efficiently searching across many PHR data items.

2.2 Searchable Encryption

One of the primary obstacles to implementing encryption is the difficulty in searching encrypted data. Commonly used encryption techniques do not support searches, as the order of the plaintext is not maintained through the encryption process.

Various researchers have proposed techniques to provide searchable encryption in [34, 59, 9, 46]. These techniques involve the data owner encrypting plaintext and a set of keywords to send to a service provider. Queriers create a "trapdoor" of keywords for which they wish to search, encrypt them, and send them to the service provider. The service provider can use the encrypted values to determine if a match occurs that would identify ciphertext that satisfies a user query. However, current solutions all require the use of a single same key for encryption for all data in the searchable domain. This restriction conflicts with our requirement to provide embedded fine-grained access control for ciphertext.

CryptDB [58] provides a solution to the problem of searching encrypted data by storing data encrypted for specific purposes. CryptDB is research software that addresses the performance limitations of accessing encrypted data stored in a database. Multiple copies of each encrypted column are stored, using different encryption algorithms, to support many requirements of common application queries. Although CryptDB does solve the access

performance issue, it relies on distinct keys that are bound to user identities. Further, CryptDB focuses primarily on transaction related queries. The Monomi [66] project uses many of CryptDB's techniques to address analytical queries, extending the CryptDB concept by splitting query processing between the server and the client. While more scalable than CryptDB for analytical queries, it still does not provide a scalable method for one-to-many encryption. Verifiable Attribute-based Keyword Search over Outsourced Encrypted Data (VABKS) [76] uses ABE to provide access control and solves the problem of searching across encrypted data in the cloud by adding encrypted keyword indexes to the ABE payload. While VABKS does provide searchability for ABE encrypted data, the technique is document-centric, requiring a defined list of searchable keywords for each ABE item, limiting its usefulness for searching across many database items.

Our ZeroVis framework approach builds on selected concepts from each of the above, and adds data owner controlled access control to better address efficient encrypted data access and overcome difficulties associated with distributed access control.

2.3 Location Privacy

The second and third parts of this dissertation address location privacy concerns. In both parts, we explore the problem of determining if a defined location is in proximity to some other location or area. There are two primary variants of the proximity detection problem. The first problem,"nearby friends", determines if two or more user locations are within a threshold distance from one another. The second approach, the focus of our research, determines if a location overlaps one or more areas defined by an authority.

Existing private proximity detection solutions generally fall into the following four main areas, each with its own drawbacks:

- Location perturbation (limited accuracy)
- Access control (limited privacy/utility trade-off)
- Private information retrieval (PIR) (lack of data owner privacy)
- Encryption (performance cost)

Also, current private proximity detection solutions, with the exception of [51], provide results based on location alone, and do not consider other attributes. Any results filtering is left to a third party that can examine attributes and learn user locations.

2.3.1 Location Perturbation and Transformation

In a location perturbation scheme users provide obfuscated or perturbed data to a Location Based Service (LBS). K-anonymity [65] is the most common technique to limit the LBS' ability to determine any user location. Kim [43, 41] proposed two different approaches to cloaking locations. One weakness of such schemes is that users must possess the ability to generate cloaked areas, and attackers can also use these techniques to analyze shared cloaked locations. Another weakness of these schemes is that the LBS only responds with answers of the same, or lower, precision of the obfuscated user location. Yiu [72] proposed a collection of schemes to strengthen the shared knowledge weakness by partitioning data using different criteria, but these techniques are still vulnerable to attacks based on a priori knowledge and brute-force attacks. Hossain [48] proposed a shear-based spatial perturbation scheme, and Yoon [73] proposed a line symmetric-based spatial perturbation scheme. Recent work in cloaking [3, 70] extends differential privacy [21] and provides greater semantic privacy protection. However, all perturbation schemes reduce location data precision.

2.3.2 Access Control

Another approach is via structured access control techniques. Bugiel [10] proposed a fine-grained Mandatory Access Control (MAC) approach for Android devices, called FlaskDroid. Li [51] proposed Privacy-preserving Location Query Protocol (PLQP) for fine-grained access to location data. PLQP allows queries to access location information while still upholding user privacy and is efficient enough to operate on mobile devices. Lu [52] proposed Secure and Privacy-preserving Opportunistic Computing (SPOC) framework for healthcare data, which requires close proximity of a patient and medical personnel to grant PHI access. Fawaz [26] proposed more fine-grained access control for sharing location data with third-party apps. Access control methods only provide binary access control with limited granularity for privacy protection, i.e. users can either grant or block access, and these approaches require a trusted third party to access location data to make access decisions.

2.3.3 Private Information Retrieval

PIR [16] allows users to issue queries to a data owner database without the data owner learning the content of the query. These techniques build private spatial indexes that utilize PIR operations, which can provide efficient spatial query processing while the underlying PIR scheme provides privacy. Hengartner [35] presents an architecture that uses PIR to hide location information from an untrusted server and uses trusted computing to guarantee that the PIR algorithms are only performing the operations as intended. Khoshgozaran [40] proposed two location privacy approaches that eliminate the need for an LBS anonymizer. Ghinita [29] proposed a PIR based technique to support Nearest-Neighbor (NN) queries without requiring a trusted third party. This approach uses cryptography to protect user locations and increases performance with data mining techniques to eliminate redundant

calculations. PIR techniques can be efficient and provide a high level of privacy guarantees, but they assume that the AOI data is public, and provide no privacy for data owners. Unlike PIR techniques, mutually private proximity detection, which is the focus of our research, requires that all AOIs defined by data owners be kept private as well as user location data.

2.3.4 Encryption

Other approaches use encryption for location data to provide privacy, requiring varying degrees of communication and computation in the encrypted space to determine proximity. Encryption techniques can be viewed as symmetric PIR, that is, PIR plus the restriction of AOI privacy, which is the main focus of our research. Although encryption introduces overhead, techniques that use it can be more resistant to attackers, even with prior knowledge. Khoshgozaran [39] proposed a spatial data encryption scheme based on a grid that uses group shared symmetric keys, allowing users to query nearby cell contents and then locally decrypt location details for items encrypted with their group shared key. Yiu [71] proposed the Metric Preserving Transformation (MPT) and Flexible Distance-based Hashing (FDH) schemes. MPT is a Mtree-based encryption technique which uses Order-Preserving Encryption (OPE) [1] to hide location information. FDH converts data into a bitmap and uses AES to encrypt each bitmap. FDH is faster than MPT, although FDH only provides approximate query results. MPT and FDH are limited in their utility value since they only support NN queries, and not "nearer than some threshold" queries. Ghinita [30] proposed a method based on Hidden Vector Encryption (HVE), and searchable encryption techniques that allow an untrusted server to determine proximity to alert zones without knowing any user's actual location. Near-Pri [55] allows users to share their specific locations once proximity (within a specified threshold) is determined.

Calderoni [11, 56] proposed a new compact data structure, Spatial Bloom Filter (SBF), to privately store AOI locations, and the Paillier cryptosystem to protect AOIs from disclosure while still allowing comparison with encoded (but unencrypted) user locations. Elmehdwi [23] proposed Secure k-Nearest Neighbors (SkNN), which uses Paillier cryptosystem and protocols to support private k-NN queries. Li [51] proposes a MPPD solution that does include fine-grained access control by using CP-ABE [4], along with Paillier cryptosystem. Li's use of Pailler is similar to the secure distance calculations of Elmehdwi's approach, and is the most relevant to our approach. We compare our framework to Li's in the experiments section. Choi [15] uses Paillier cryptosystem and Order Preserving Encryption (OPE) [1] to detect proximity between proximity zones. Sedenka [67] also uses Paillier crytosystem, but incorporates garbled circuits to define three protocols to privately calculate distances between any two points using different coordinate systems on a spherical surface.

Kim [42] proposed Hilbert Curve Transformation (HCT), which encodes locations using a Hilbert Curve and then encrypts the result using AES. Wang [68] proposed an approach to perform a geometric range search on encrypted spatial data. Encryption techniques are promising and can offer good privacy guarantees for both users and data owners, but at a cost. Encryption requires computation overhead which can be a drawback for devices with limited processing power. Our approaches combine several encryption techniques, including CP-ABE, HVE, and Paillier encryption to address the MPPD problem.

Chapter 3

Cloud Database Confidentiality with Fine-grained Access Control

3.1 Problem Definition

In this work [62], we address the difficulties encountered with providing confidentiality for users from different categories for data outsourced to a CSP. Consider a database D, with tables T_1 .. T_i. Each Table contains rows R_1 .. R_j, each with columns C_1 .. C_k. Clients access the database contents as data owners/providers (DO), users, or in both roles. Data owners store data in the database (INSERT, UPDATE), and users retrieve data from the database (SELECT). In a database that uses client-based encryption to protect stored data, clients access data in one or more columns (C_1, C_2, ..., C_k) from one or more rows (R_1, R_2, ..., R_j) from one or more tables (T_1, T_2, ..., T_i). Data owners encrypt data before storing it in the database and users must decrypt data after retrieving it from the database. In this model, the database only stores encrypted versions of protected cells and never sees the plaintext version of the data. The primary problem with this approach is in

the difficulty of generating and managing the keys to encrypt and decrypt data. Data owners and users must share keys to access data, and the number of keys grows with a higher level of desired fine-grained access (i.e. a need for more encryption partitions.)

3.1.1 Running Example

Difficulties with balancing data protection and ease of access are common in medical data collection. Consider the following scenario. Four research teams, A, B, C, and D, need patient data. Table 3.1 shows four research teams, along with the specific treatments they are studying and the general context of their work. The primary challenge to be addressed is to obtain current data that is pertinent to their research, while complying with HIPAA rules and patient constraints. Patient constraints allow a patient to control who can access her data, such as researchers, medical service providers, and next of kin. A patient can submit her data with constraints, such as "allow authorized cancer researchers to access my data". Each team's attributes are used to determine whether patient constraints are satisfied for each query.

Uncoordinated independent requests by each team to individual patients via different medical providers is clearly tedious and impractical. On the other hand, storing and accessing all data via a centralized, trusted 3rd-party creates a single point of trust as well as failure, which can only be partly mitigated by hierarchical grouping of providers and consumers.

Given our running example, assume that a patient received treatment at an oncologist's office. The patient specified that the data to describe and record the visit is saved with the following access policy:
"treatment='Z51.11' AND (context=cancer OR context=tobacco use)" (i.e. only users that possess the treatment attribute with a value of Z51.11[1] and

[1]Z51.11 is the ICD10 code for "Encounter for antineoplastic chemotherapy", which also

Table 3.1: Data Access Example: Research teams and contexts of interest

Team	**Treatment**	**Research Context**	**Description**
A	Z51.11	Cancer	Effectiveness of different treatment
B	E66.09	Nutritional health	Impact of obesity on heart disease
C	Z51.11	Tobacco use AND Heart disease	Impact of lifestyle and heart disease on cancer treatment effectiveness
D	Modified Diet	Nutrition	Measurable benefits of various diets

the context attribute with either the values of cancer or tobacco can access her data.)

Data is stored encrypted in a database, but can only be decrypted by users who possess attributes that satisfy the access policy. In this example, all research teams could retrieve any encrypted database row (including the patient data encrypted using the access policy from above). However, only research teams A and C could decrypt the data since their attributes (treatment and context) match the saved access policy. (Although research teams B and D can retrieve the encrypted data they cannot decrypt it and it is therefore unusable to them.)

corresponds to the ICD9 code V58.11.

3.1.2 Building Blocks

We built the ZeroVis [62] framework on two primary building blocks, Ciphertext Policy Attribute Based Encryption (CP-ABE), and CryptDB. Each component brings desirable features to ZeroVis, but neither one solves our problem alone.

Ciphertext Policy Attribute Based Encryption A CP-ABE scheme provides fine-grained access control over data [4]. CP-ABE associates a user with a set of descriptive attributes to generate the user's secret key, SK. Data are encrypted under an access policy such that only users whose attributes match the access policy can decrypt the data. Existing ABE schemes are generalizations of Identity Based Encryption (IBE) schemes[7]. IBE uses only one attribute, the identity of the receiver, as opposed to ABE systems that support using multiple attributes simultaneously. There are 2 main ABE alternatives, defined by the location of the access policy. Key Policy ABE (KP-ABE) embeds the access policy in the user's private key[32]. The other alternative is CP-ABE, which embeds the access policy in the ciphertext[4]. The primary difference between the two schemes is that KP-ABE gives control over who can decrypt data to the key generator, while CP-ABE ensures that the encryptor (data owner) retains control over who can decrypt her data[4].

To encrypt a message M using CP-ABE, the encryptor provides an access policy which is expressed as a boolean expression containing selected attributes and values for M. Figure 5.2 shows the access policy presented earlier (section 3.1.1) in a tree structure. The message is then encrypted based on the access structure, T. Decryptors generate SK based on their attributes. A decryptor is only able to decrypt ciphertext, CT, when her SK satisfies the access policy used to encrypt the message. Unauthorized users cannot decrypt CT even if they collude and combine their disjoint attributes.

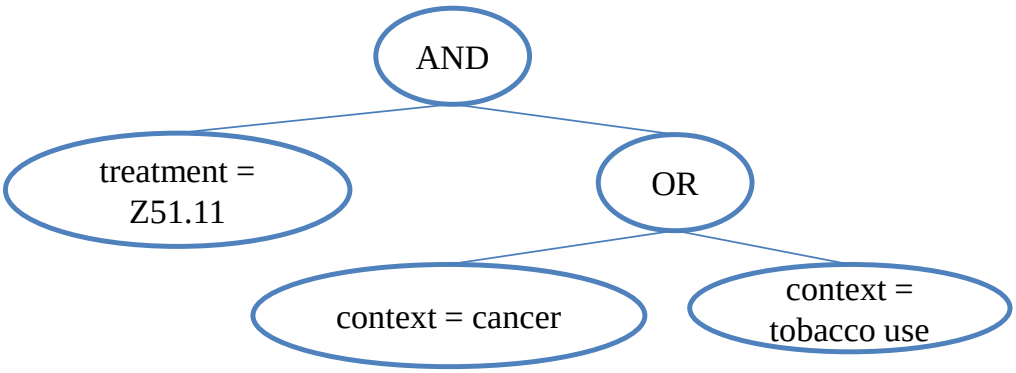


Figure 3.1: CP-ABE Access Tree

CP-ABE defines the following four essential functions:

1. Setup(): Input security parameter, output public parameter (PK), for encryption, and master key (MK), to generate user secret keys.
2. Encrypt: Input message M, access structure T, public parameter PK, output ciphertext CT.
3. KenGen: Input set of user's attributes SX and MK, output secret key SK for SX.
4. Decrypt: Input CT, SK. If SK satisfies access structure in CT, return M, else return NULL.

CP-ABE works well for encrypting individual shared data where the file's name or identifier is known, but there is no provision for searching ciphertext, thereby making CP-ABE alone insufficient for database queries.

CryptDB. CryptDB is a DBMS that provides confidentiality for data stored on an untrusted database server [58]. The system provides near-transparent confidentiality by intercepting database queries and rewriting

them in such a way as to execute over encrypted data. Decryption for consumption never occurs on the server, only at the trusted proxy. CryptDB also incorporates an encryption strategy that can adjust the encryption level of each column based on user queries. At runtime, the CryptDB proxy analyzes each query and determines the encryption needs based on the query components. The proxy will either then map each query component to an encrypted data item or request an encryption layer adjustment. All data is initially stored by CryptDB after being encrypted into multiple layers, with each layer being encrypted with one of six encryption methods. CryptDB can encrypt each data item using Random (RND), Deterministic (DET), Order Preserving Encryption (OPE), Homomorphic Encryption (HOM), Join and OPEjoin, Word search (SEARCH) encryption. For instance, a data item may be encrypted with Join encryption method, encrypted again with an OPE encryption method, and then encrypted once again using a RND method. CryptDB selects the encryption methods and layer construction based on the data item's data type. The resulting value is called an "encryption onion" with three layers. CryptDB will only "peel" an onion layer (decrypt the outer layer) if a query requires an inner layer to successfully complete. This dynamic ability to alter encryption layers gives CryptDB the flexibility to maintain confidentiality while still responding to query requirements. The database server peels onion layers with user defined functions, and will never remove the innermost layer that would expose the original plaintext. The database server executes supplied queries over encrypted data and then returns the data to the proxy. The CryptDB proxy then decrypts the data and returns the plaintext to the client. The server never sees the plaintext.

Popa, et al[58], report that the additional overhead only modestly reduces the DBMS performance. When evaluating CryptDB running on Postgres, the TPC-C benchmark demonstrated a 27% throughput reduction over an unmodified Postgres environment. The authors note that one of the advantages

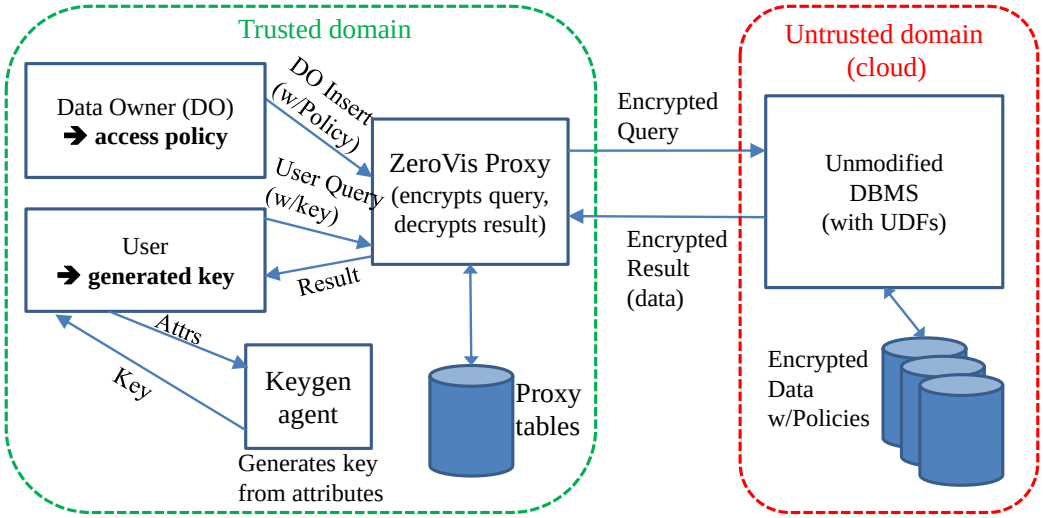


Figure 3.2: ZeroVisibility Cloud Framework.

to the CryptDB approach is that implementing CryptDB does not require DBMS engine changes. All of the CryptDB functionality is implemented in the proxy and by supplying user defined functions, making it portable to new database engines with a minimal amount of effort.

Although CryptDB does provide the ability to select and search encrypted data on an untrusted sever, it still requires user-based encryption keys. CryptDB must rely on an external authority to enforce key management, including authorizing multiple users to decrypt an owner's data. This reliance on an external authority takes the power over who can access data away form the data owner.

3.2 ZeroVis Framework

3.2.1 Framework Overview

To overcome the problems described in the previous section, our approach integrates CP-ABE with the ability to search across encrypted data, e.g. as provided in CryptDB, to synthesize a solution that supports single data owner encryption accessible by multiple users for data stored in an untrusted environment, along with the ability to efficiently retrieve the data without decrypting in the cloud.

Figure 3.2 shows the ZeroVis framework. The core of our framework is the ZeroVis proxy which is responsible for encrypting data and queries and decrypting query results. The data owner submits data along with access policies through ZeroVis Proxy which encrypts the data via CP-ABE and searchable encryption and stores the encrypted data through an unmodified DBMS. A user submits a query along with a pre-generated secret key, SK, (generated from the user's attributes) through the ZeroVis proxy which encrypts the query. The DBMS returns encrypted results of the query to the ZeroVis proxy, which decrypts the results and returns the plaintext to the user.

One additional requirement of a complete framework in a production environment is an Attribute Authority (AA). The AA is responsible for authorizing users, and managing attributes associated with those users. The framework depends on the AA to supply authenticated attributes for each authenticated user, and to prevent unauthorized users from submitting queries through the framework. Users can submit queries directly to the untrusted DBMS, but without the necessary master key from the AA, decryption attempts are unsuccessful.

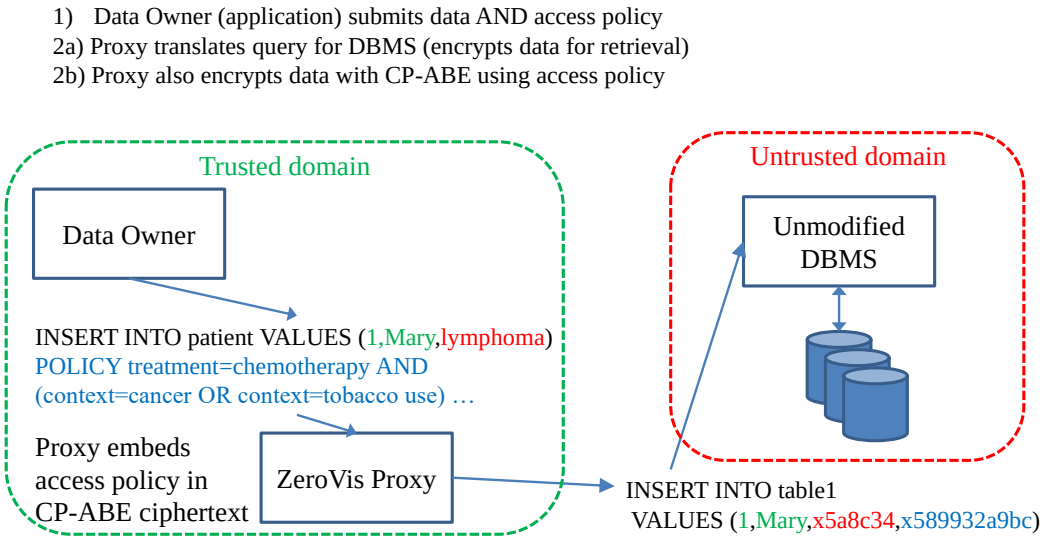


Figure 3.3: Submitting Data (INSERT)

3.2.2 Data Insertion and Encryption

To encrypt data, the data owner provides the trusted proxy with the plaintext data and an access policy. Figure 3.3 shows the data flow with an example INSERT query. The trusted proxy encrypts the plaintext data, translates the query components into their encrypted counterparts (for query elements that are stored encrypted in the DBMS), and submits the encrypted payload, along with the embedded access policy, to the DBMS. Notice in Figure 3.3 there are 2 ciphertext values. The first represents existing CryptDB encryption and the second depicts the new CP-ABE ciphertext added by ZeroVis.

3.2.3 Data Retrieval and Decryption

To decrypt data, a user must first generate a secret key, SK, based on her attributes. In most implementations, a trusted attribute authority will generate a key for each identity upon new user registration. The user provides a set of descriptive attributes, SX, such as treatment and context interest areas

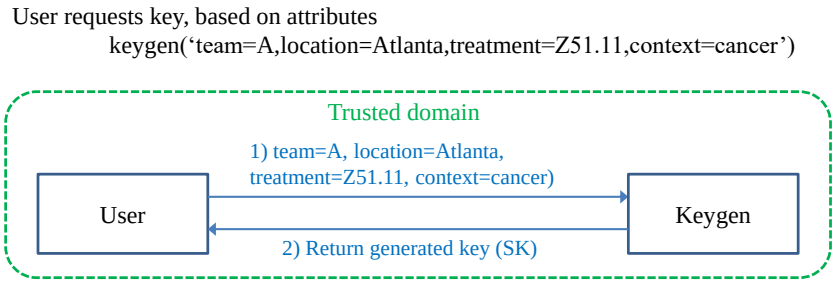


Figure 3.4: Generating the Data Access Secret Key

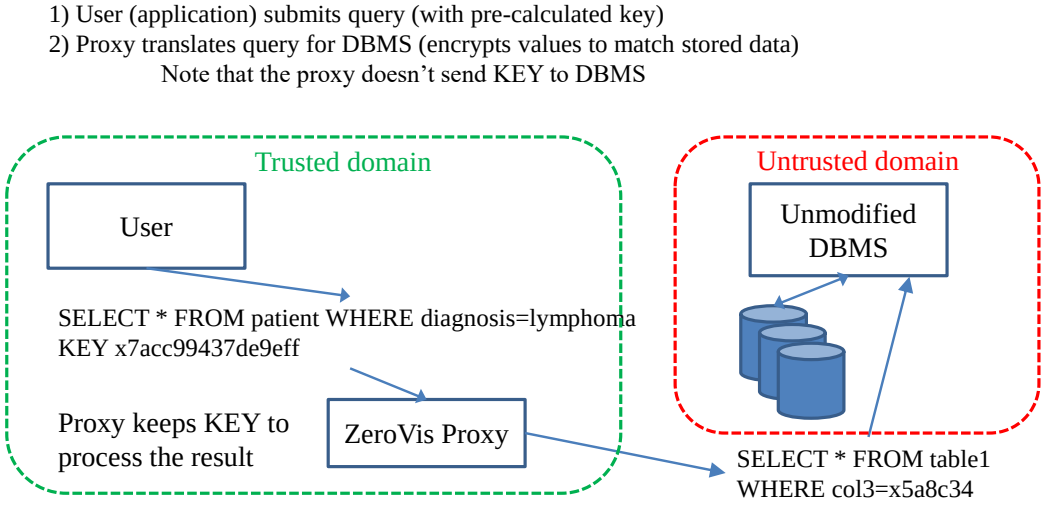


Figure 3.5: Retrieving Data (SELECT request)

(for our running example). Attributes can describe an entity's state, status, or authorized interest areas. The attribute authority generates SK based on the supplied SX and returns SK to the user on demand. For example, a research team member may possess attributes:

"treatment=Z51.11, context=cancer".

Figure 3.4 shows the process of the attribute authority generating a secret key, SK, based on the supplied attributes and then returning SK to the user.

The user then presents SK (generated by the attribute authority) to the trusted proxy when attempting to access encrypted data. The trusted proxy

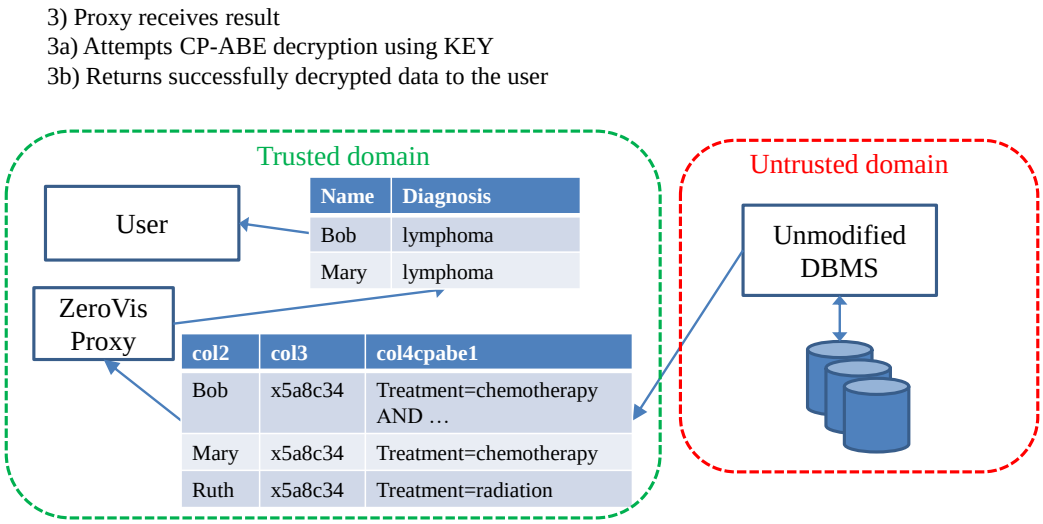


Figure 3.6: Retrieving Data (SELECT response)

translates the supplied query elements into their encrypted counterparts (for query elements that are stored encrypted in the DBMS), and submits the query, depicted in Figure 3.5. The proxy then translates the returned data from the encrypted state, CT, as stored in the DBMS, depicted in Figure 3.6, into plaintext state, M, for the application. The CP-ABE decryption algorithm will only return plaintext message, M, when the supplied SK satisfies the data's embedded access policy that was provided by the data owner. If the supplied key does not satisfy the access policy the proxy simply returns a null value.

The process of modifying data (UPDATE) is essentially a combination of a data retrieval operation followed by a data submission operation. While the process of updating data is straightforward, the implementation of the framework would need to ensure that updates are well-behaved and do not allow unauthorized data or policy modifications. Users updating data must possess SK to retrieve data and an access policy to encrypt changes. Traditional access controls would be necessary to limit data and policy updates to

authorized users.

3.3 ZeroViz Client Walk-through

ZeroViz supports both interactive clients through a shell prompt and existing applications through a connection to the proxy. Both client types require that users register with an Attribute Authority (AA). This section shows the steps a ZeroVis client follows to register with the ZeroVis system, submit, and query data.

3.3.1 Attribute Authority Registration

1. Create an account with an attribute authority that ZeroVis trusts (set up in ZeroVis configuration.)

2. Provide each user's attribute values to the attribute authority. The attribute authority authenticates each user's identity and stores attribute values for each identity.

3. The attribute authority also uses the supplied attributes and values to generate a CP-ABE key, calling the CP-ABE kegen() function.

   - For example:

   ```
   keygen(team=A, location=Atlanta,
          treatment=chemotherapy, context=cancer)
   ```

4. The attribute authority stores the generated key for each validated user profile. Each user may possess several profiles, each defined by a distinct set of attributes and corresponding values.

3.3.2 INSERT data

The prerequisites for INSERTing data are:

- Column names and data
- Access policy expressed as a boolean expression of attributes and corresponding values
  - For example: "treatment=chemotherapy AND (context=cancer OR context=tobacco use)"

1. Launch ZeroVis
   (a) Command to start the ZeroVis proxy:

```
/path/to/cryptdb/bins/proxy-bin/bin/mysql-proxy \
    --plugins=proxy --event-threads=4 \
    --max-open-files=1024 \
    --proxy-lua-script=$EDBDIR/mysqlproxy/wrapper.lua \
    --proxy-address=127.0.0.1:3307 \
    --proxy-backend-addresses=localhost:3306
```

   (b) Command to start an interactive ZeroVis session:

```
mysql -u root -pletmein -h 127.0.0.1 -P 3307
```

2. Enter desired SQL query
   (a) INSERT INTO patient VALUES() POLICY ...
   - For example:

```
INSERT INTO patient VALUES (1,Mary,lymphoma)
POLICY treatment=chemotherapy AND
(context=cancer OR context=tobacco use)
```

The ZeroViz proxy does the following for INSERT (see Figure 3.3):

1. Rewrites the supplied query, replacing table and column names with obfuscated names and encrypting literal values when storing data in encrypted columns. The encryption method used for each column corresponds to the current encryption level for that column, maintained by CryptDB.

2. Additionally encrypts each "CP-ABE protected" column with CP-ABE using the supplied policy, and adds the resulting ciphertext to the INSERT query.

3. Submits the rewritten query to the database.

3.3.3 SELECT data

1. Launch ZeroVis (same commands as above)

2. Enter desired SQL query

   (a) SELECT * FROM patient WHERE criteria KEY keyFromAA ...

   - For example:

     ```
     SELECT * FROM patient WHERE diagnosis=lymphoma
     KEY x7acc99437de9eff
     ```

The ZeroViz proxy does the following for SELECT (see Figure 3.5 and Figure 3.6):

1. Rewrites the supplied query, replacing table and column names with obfuscated names and encrypting literal values when searching encrypted columns. The encryption method used for each column corresponds to the current encryption level for that column, maintained by CryptDB.

2. Submits the rewritten query to the database.

3. Uses the CP-ABE key (from the attribute authority) to decrypt columns encrypted with CP-ABE.

4. Discards rows that contain any field for which CP-ABE decryption fails.

5. Returns all remaining rows to the user.

3.3.4 Implementation

Our test implementation of ZeroVis was built on the architecture described above. The user issues queries to the ZeroVis proxy. The ZeroVis proxy re-writes each query and submits it to the mySQL database server. We built the ZeroVis proxy by modifying the CryptDB proxy, which was built by modifying mysql-proxy. Both CryptDB and ZeroVis can be implemented using other proxy software and any DBMS the chosen proxy supports. Both the ZeroVis proxy and the MySQL database server run on computers running Linux. ZeroVis supports both interactive clients through a shell prompt and existing applications through a connection to the proxy. Both client types require that users register with an attribute authority.

We implemented the ZeroVis framework by integrating CP-ABE into the CryptDB proxy. CryptDB provides query re-writing and capability to search across encrypted data. The addition of CP-ABE as a new encryption method within CryptDB gives the framework one-to-many encryption capability. The first change to CryptDB was to create a new column for each protected column. CryptDB normally creates 2 or 3 columns to store encrypted data using different methods to support different types of queries. The new column for each plaintext column stores the CP-ABE ciphertext. We added a new encryption layer, ABE, to each onion definition, added a new ABE security level, and added a new class to handle CP-ABE encryption and decryption operations. The new class uses cpabe-toolkit functions to encrypt

and decrypt data. We modified the CryptDB proxy query re-writing code to replace requested columns with CP-ABE columns. We retain the CryptDB obfuscated column names in the queries to allow the database to select data using searchable data. The database then returns only CP-ABE encrypted data. The proxy attempts to decrypt each column and returns successfully decrypted data to the client.

With the new functionality in place to handle CP-ABE, we extended the proxy to fetch the user's CP-ABE SK, based on the MySQL database user id. The private key will be provided by the attribute authority in more robust implementations. Additional modifications to the proxy also fetch and store the current user's default access policy for CP-ABE encryption operations. The ZeroVis system currently creates CP-ABE ciphertext for every encrypted column. The CP-ABE encryption uses the current user's access policy. Decryption uses the current user's SK, previously generated using the CP-ABE keygen() function. Future work will extend the supported SQL syntax to allow users to optionally provide access policies with every query.

3.4 Performance Results

Experiment Setup. Our performance assessment is based on a straightforward CP-ABE addition to CryptDB as described above. Our goal was to determine the additional overhead CP-ABE added to the existing CryptDB implementation. We created multiple copies of test databases, all based on subsets of the TPC-C[18] benchmark database. Test databases of different sizes were built by altering the number of rows in the item, warehouse, and district tables, all based on cardinality relationships defined in the TPC-C specification. The resulting 5 test databases DB-a, DB-b, DB-c, DB-d, DB-e have row cardinality of 1912, 2975, 7156, 18622, 35756 respectively, which are

approximately increasing in a logarithmic scale. We created sets of queries, both single row and multiple row returned sets, to assess the general performance of the ZeroVis framework. The queries were simple, single table INSERT statements to load varying size subsets of the TPC-C database, and single table SELECT statements to retrieve 1 row (150 SELECTs) and sets (150 SELECTs) from the item, stock, and customer tables. The SELECT queries to retrieve sets of rows were randomly generated to select a range from the domain of each table. The test sever had an Intel Core 2 2.0 GHz(x2) processor with 3GB RAM running Ubuntu 13.10. The client/proxy computer had an Intel Core i7 2.4 GHz(x8) processor with 16GB RAM running Ubuntu 13.10. The two computers were connected via a 100Mbit/s Ethernet connection.

Results. Adding CP-ABE results in an additional encryption operation for each protected column, adding substantial observed space and computation time overhead. CryptDB, without CP-ABE, is approximately 26% slower (throughput loss) than native MySQL [58] when running the TPC-C benchmark. Encryption and decryption times are linearly related to the number of leaf nodes in the CP-ABE access policy. According to Bethenourt et al, [4], their implementation of CP-ABE took approximately 0.5 seconds to encrypt a payload with 20 policy leaf nodes, while only taking 0.04 seconds to decrypt. One reason why the encryption operation is so much slower is that it includes parsing and processing the provided policy. The decrypt operation does not directly interact with attributes. Generating the key, based on supplied attributes, is a separate function that must be completed prior to any decryption attempt.

Figure 3.7 shows the resulting database size (in MB) of the 5 test databases with varying row cardinality (approximately increasing in a logarithmic scale) for CryptDB (without CP-ABE) and ZeroVis (with CP-ABE) respectively.

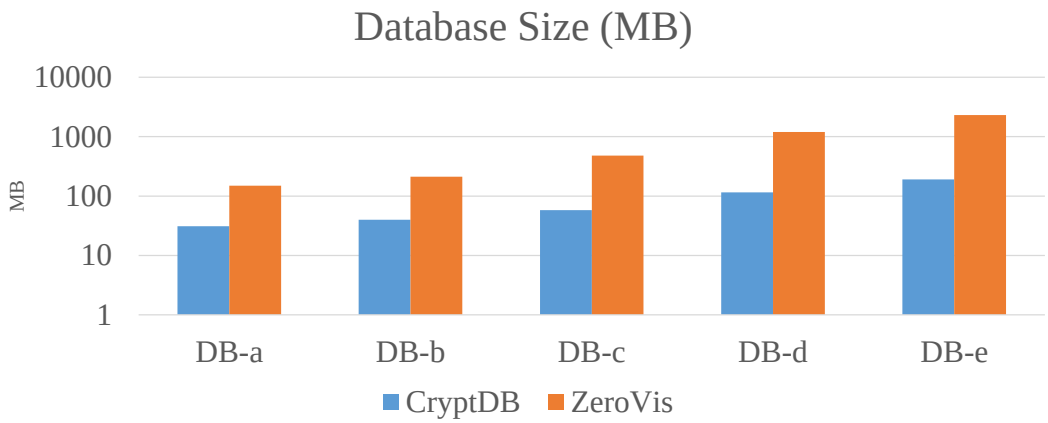


Figure 3.7: Resulting DB Sizes for test DBs (of logarithmically increasing row cardinality)

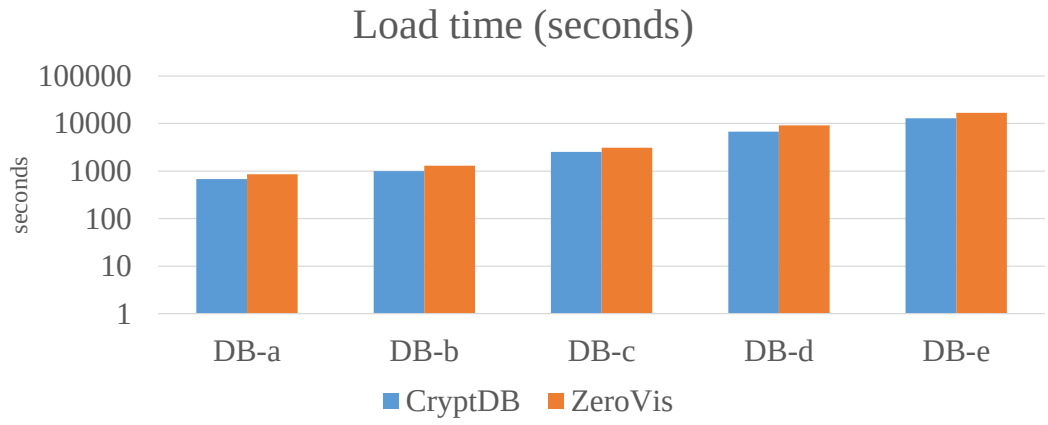


Figure 3.8: Database load time

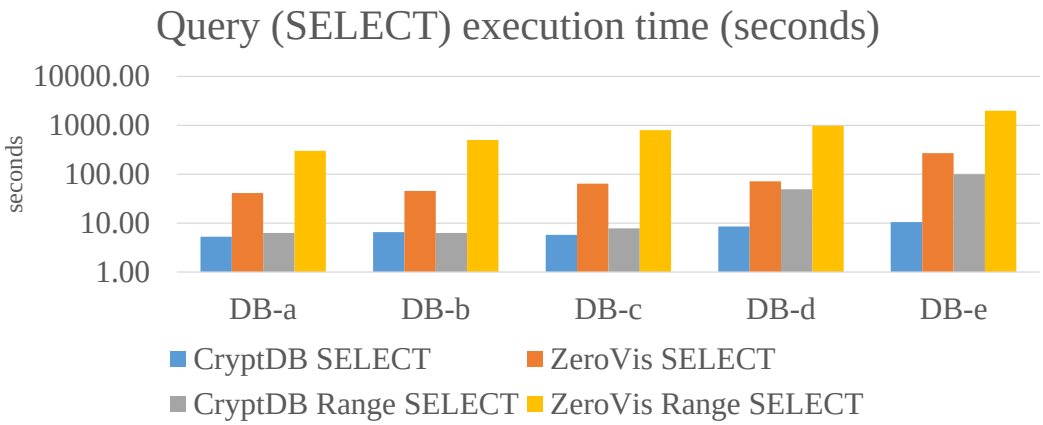


Figure 3.9: Database Query time

As the figure illustrates, the overhead incurred by ZeroVis increases linearly with the row cardinality. The current implementation stores a complete CP-ABE ciphertext payload for every protected database column, which includes the access policy and the encrypted data. Future work will explore reducing redundancy through consolidating CP-ABE access policies which we expect will significantly decrease the overhead.

Figure 3.8 shows the load time for each database instance. The ZeroVis computational overhead is a result of the additional CP-ABE calculations. As mentioned above, the current test ZeroVis implementation constructs the access policy tree for each column, even if all columns share the same policy. It is expected that reducing CP-ABE access policy redundancy will also reduce computational overhead for future ZeroVis framework versions and result in ZeroVis performing more closely to CryptDB.

Figure 3.9 shows times for queries that return single rows, and sets of rows (range queries). We submitted 300 SELECT queries for each database instance, 150 distinct queries and 150 range queries. The queries were scaled to consider the range of data stored in each database (randomly generated to exercise the full range of data in each table). Queries use both indexed

and non-indexed criteria. The disparity between CryptDB and ZeroVis performance for range queries is due to ZeroVis' current larger data storage requirements. Additional tests with the proxy and sever running on a single machine showed that network costs were not responsible for the higher overhead of ZeroVis. The queries in our test returned most of the columns from each table, requiring CP-ABE decryption operations for each column. While decrypting multiple columns is normal expected behavior, the redundancy of storing and transporting multiple copies of the access policy for each column increases the workload.We believe reducing redundant CP-ABE operations and normalizing the access policy storage technique will reduce ZeroVis' computational overhead and additional costs of the framework, resulting in performance closer to CryptDB than the current ZeroVis implementation.

3.5 Conclusions

In this section we showed how combining CP-ABE with encrypted data searching addresses the problem of storing and retrieving confidential data from an untrusted environment, while giving the data owner control over who accesses her data. While other frameworks provide some of these capabilities, ours is the only one to our knowledge that accomplishes this without relying on traditional key management techniques. Our framework is the first to specifically address the need for one-to-many encryption in a database environment, which requires support for efficient queries across encrypted data. This dissertation describes the initial ZeroVis framework implementation. Future framework changes are necessary to create a more production viable framework. We discuss future work in Chapter 6.

Chapter 4

Mutually Private Proxmity Detection Evaluation

4.1 Problem Definition

This second work [63] focused on a critical evaluation of proposed techniques to provide MPPD to provide guidance that implementers of MPPD solutions can use to select a "best fit" technique for a specific environment. We focused on MPPD techniques that use encryption to maintain precise location data to maximize the accuracy of any proximity query results. Our goal was to survey likely candidates, select a representative group from proposed techniques, implement the selected approaches, and evaluate their relative performance. In addition to producing useful comparative performance statistics, we also wanted to provide a basis to develop the extended framework we will describe in Chapter 5.

4.1.1 System Setting

We evaluated five encryption-based MPPD methods, and selected three to implement and assess. In our comparative assessment of techniques, we consider the following architecture components:

1. Data Owner/Provider (DO) - Defines the AOIs
2. Service Provider (SP) - Computation and location services
3. User - User with a location sensing capable device

Each of the methods differs in the techniques used to provide location privacy and in how they define architecture component responsibilities. Each method defines a trusted component to manage encryption keys and at least one computation component to handle much of the computation load. The distribution of responsibilities among components impacts how well each one performs. Method descriptions in Section 4.2 explain how each method defines responsibilities.

The methods we implement define AOIs either as circles, each with a defined radius, or as a collection of grid cells that comprise each AOI. Circles are more generic but collections of cells allow for non-uniform AOI shapes. Figure 4.1 shows 2 different ways to define AOIs. - circles and defined grid cells. In the figure there are 3 defined Areas of Interest (AOI). The user, Alice, is in proximity to AOI 3, (i.e. her location overlaps the region define by AOI 3).

Figure 4.2 shows the general architecture for MPPD. One or more data owners defines, encrypts, and sends AOIs to the service provider. The user initiates the query by asking "am I near an AOI?", and sending an encrypted location to the service provider. Further, all algorithms can be implemented as a request-response model to support on-demand queries, or a publish-subscribe model to support push notifications. The specific steps each protocol takes to detect user/AOI proximity differs, but all use the same components in Figure 4.2.

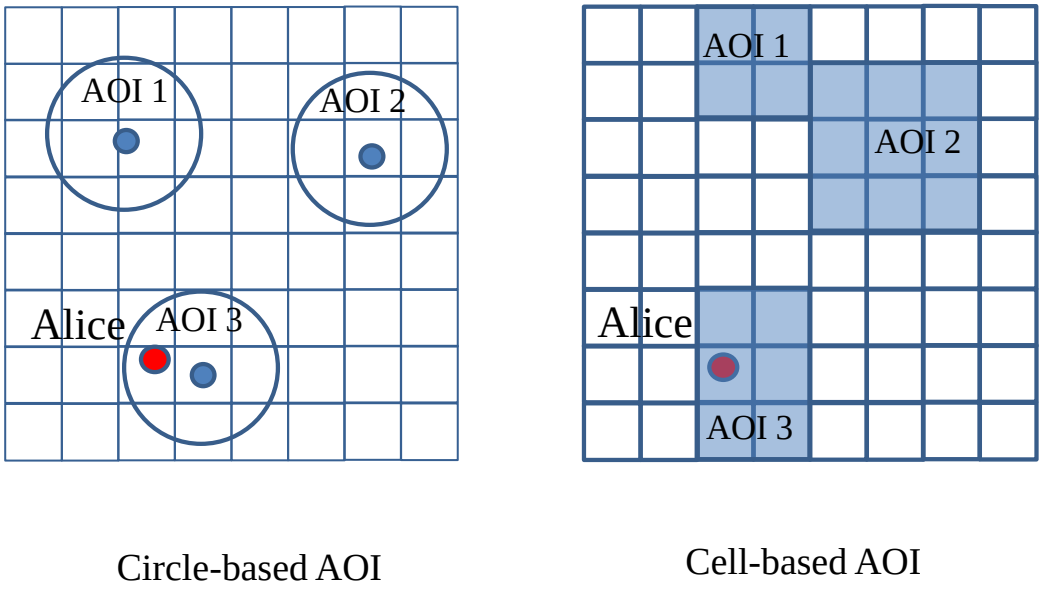


Figure 4.1: Mobile user and 3 Areas of Interest (AOI), using circle-based and cell-based AOI definition

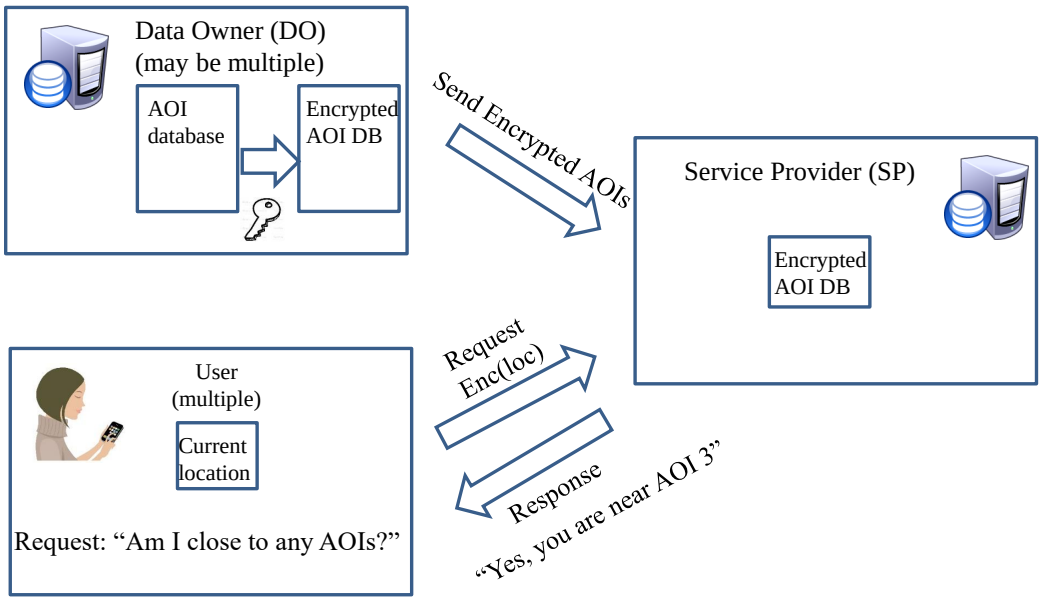


Figure 4.2: Mutually Private Proximity Detection Architecture

4.1.2 Problem Statement

We examine five MPPD approaches that can be deployed to existing mobile devices. The approaches we analyze allow users to send location information to a service provider and receive AOI proximity notification without divulging their location to the service provider. At the same time, these approaches allow data owners to publish AOI information for users to consume when they are within a specified distance from AOIs without publishing AOI locations in advance. We define proximity to an AOI to be the condition in which a specific location is within the area defined by the AOI boundaries. Proximity can be determined when a location is closer to the center of an AOI than some defined threshold or that the location is within a grid cell included in an AOI. Data owners can define AOIs of any size. Any of the approaches in this dissertation can be extended to support the case in which a single AOI represents a specific user. In this case, the data owner would be a user and the techniques would provide user to user proximity detection. However, the focus of our dissertation is to determine user proximity to one or more AOIs, and techniques designed for user to user proximity is outside our scope. System designers deploying MPPD have several options available to them. Our analysis results can help determine the most appropriate approach, depending on how input data (size or number of AOIs and size of the grid of AOIs) changes over time.

4.1.3 Security and Privacy Goals

Proximity alerts, by definition, disclose information. The primary goal of the protocols we implemented is to provide proximity detection to defined AOIs without divulging either the user or AOI location. In that way, the location privacy of user and data owner data is maintained. Of course, some information will always be divulged when a user is in proximity to one or

more AOIs. A positive service provider response ("Yes, Alice, you are close to AOI 3") divulges the general location of that AOI to Alice, and the service provider learns only that Alice is near AOI 3. Note that the service provider does not learn Alice's location or any AOI location information - only that Alice is within the boundaries of AOI 3. The amount of information divulged to users is directly related to the size of the AOIs. Proximity to large AOIs only divulge general information, while proximity to small AOIs would result in a finer granularity of location information divulged. Our analysis does not consider the amount of location information that could be divulged among colluding actors.

We consider the three privacy guarantees listed below for each of the methods considered. Protecting data from unauthorized disclosure is handled by the security guarantees of each cryptosystem.

- **User location privacy** - User location is never divulged to other users, any service providers, or data owners. The data owner can determine if a user is in proximity to any AOI, with accuracy dependent on the AOI size.

- **Data Owner/Provider privacy** - Data owner defined AOIs are never divulged to any user unless that user is near an AOI, and never to any service provider. Users can infer AOI location when they are near one or more.

- **Query privacy** - Locations sent to the service provider and responses returned to users do not provide any information either party could use to determine current or historical location.

While our assessment focuses on privacy, it is important to address the security guarantees of each method. Without security guarantees against specific attacks, there is no basis for discussing privacy guarantees. The

Table 4.1: Methods and techniques

Method	**Description**	**Technique**
SBF	Spatial Bloom Filter	Bloom Filter + Paillier
SkNN	Secure k-NN and range query	Paillier
HCT	Hilbert Curve Transformation	Hilbert Curve + AES

methods we chose to implement use two different types of encryption to provide confidentiality. Table 4.1 summarizes the chosen approaches and each one's techniques.

SBF and SkNN use the Paillier cryptosystem to encrypt locations. The data owner creates key pairs, and only the data owner can decrypt data. Paillier is a probabilistic asymmetric key encryption algorithm and has the indistinguishability under chosen plaintext attack (IND-CPA) stated as: an adversary operating in polynomial time has the same probability of success as simple random guessing. Suppose the adversary chooses two plaintexts, and we select one of the plaintexts randomly and return the ciphertext of our chosen plaintext to the adversary. The adversary guesses which plaintext corresponds to the ciphertext. We say the encryption is IND-CPA if the attacker can achieve no better success probability than 1/2 + some negligibly small number. Using the Paillier cryptosystem, an adversary with reasonable computer processing power may obtain knowledge of some ciphertext, but will not be able to obtain any useful knowledge of the corresponding plaintext [53].

HCT uses AES to encrypt its location and index tables. The data owner creates the key and only shares it with authorized users. AES, when used in an appropriate mode, such as Cipher Block Chaining (CBC) or Counter (CTR), also has the IND-CPA property [5].

4.2 Method Description

4.2.1 Overview

We selected five MPPD methods to examine, and chose three of those to implement to evaluate their scalability and use the performance comparison to suggest a best approach for specific requirements, such as a large number of AOIs or a dynamically changing coverage area size. This chapter describes the differences between the implemented methods, their security and privacy guarantees, and how well sample implementations perform with varying computation loads (varying number of AOIs, size of AOIs, and size of the grid on which locations are mapped). One of the methods was not designed to solve our problem per se, but offered a framework that presented an opportunity to implement modifications to align it to our goals. The SkNN approach was originally proposed to solve a problem in a different domain. The original protocol determines the distance between vectors of attributes, as opposed to calculating the distance between two points. For example, such an approach is useful in determining the closeness of database rows. We modified the original proposed approach to address the problem of proximity detection while maintaining the original intent of protecting mutual privacy.

4.2.2 SBF (Technique - Bloom Filter)

Calderoni [11, 56] proposed a new compact structure, SBF, to privately store location information, and two protocols to determine AOI intersection. One protocol is for two-party setting involving only a user and the data owner, while the other protocol is for a three-party setting, in which the data owner outsources communication with the users to a service provider. The SBF differs from a normal Bloom filter (BF) in that the SBF can be constructed over multiple sets, where the BF is constructed only over a single set. This

difference is desirable since AOIs may be disjoint and require representation as separate sets of grid cells. Further, the probability of a false positive SBF result depends on the order in which set membership is assessed. Accuracy can be increased by prioritizing the order in which AOIs are assessed. SBF depends on Paillier cryptosystem, as it is additively hormomorphic. This property supports calculating the sum of two encrypted numbers, and the product of an encrypted number and an unencrypted number. The data owner can send an encrypted SBF of AOIs to any user to perform AOI proximity detection calculation without disclosing AOI locations. SBF requires locations (latitude/longitude) to be mapped to distinct cells in a predefined grid. The precision of the location mapping depends on the mapping function. A single grid cell can represent any area size, but in our implementation we use a mapping function that uses longitude and latitude values expressed with an accuracy of three decimal points. This results in a grid cell representing approximately 111m x 111m (at the equator). In practice, any precision can be represented, as well as grids that represent any subset of the entire Earth's surface. The only requirement is that all users and the data owner use the same grid definition (and mapping function.) SBF defines the data owner as the trusted entity and the service provider as an untrusted computation component. Algorithm 1 lists the SBF protocol steps.

Privacy

SBF guarantees the privacy of all parties. The two-party protocol guarantees the correctness of the result for the data owner (within a stated probability), and user location privacy. SBF also provides privacy for the data owner in that user does not know (and cannot derive) the coordinates of any AOI. The three-party protocol introduces the service provider to handle communication and computation. The service provider cannot decode user's location unless the service provider gains access to the grid defined by the data owner and

Algorithm 1 SBF Three-party Protocol

1: The data owner creates an SBF for the AOIs, and Paillier public/private keys
2: The data owner encrypts the SBF using Paillier public key and sends it to the service provider
3: The data owner sends k hash functions and the conventional grid to the user
4: The user creates an SBF for current position and sends result to the service provider
5: The service provider computes entrywise homomorphic product of the SBFs
6: The service provider shuffles calculation results and sends result to the data owner
7: The data owner decrypts the SBF and counts the number of non-zero entries. Using this information, the data owner can determine if user is located in any AOI, and if so, which one.
8: The data owner sends result back to the user (which AOIs user is near, if any)

provided to the user. As long as the data owner keeps the grid definition secret from the service provider, all guarantees from the two-party protocol hold for the three-party protocol. Further, SBF provides query privacy since the data owner only sees obfuscated results of the entrywise homomorphic product of the data owner's and user's SBFs. The data owner can keep history of how many non-zero entries are in the SBF product that user sends, but there is no direct discernible correlation between user's obfuscated SBF product and user's true location. Although the data owner can determine if user is in proximity to an AOI (and if so, which one), due to the user's randomizing the results of the SBF product, each request sent to the data owner is unique (even if user is at the same location.)

4.2.3 SkNN (Technique - Homomorphic Encryption)

Elmehdwi [23] proposed two methods, a basic protocol that is not fully secure, and a more complex protocol that strengthens security guarantees. SkNN splits the computation between two components, defining the data owner as an untrusted computation component, and the service provider as a trusted computation component. Algorithm 2 provides SkNN protocol details. We made two material changes to the original SkNN approach. First, we changed the distance calculating algorithm. Our new algorithm takes the (x,y) coordinates of the user and a vector of (x,y) coordinates, along with and a threshold value for each defined AOI, as input. This differs from the original SkNN protocol that takes two vectors of attributes as input. The updated SkNN returns the squared distance between the user's location and each AOI's center, for all distances that are smaller than the squared threshold value. Second, the original algorithm only returns the top k matching points, but our implementation allows the service provider to return any number of nearby AOIs. These changes were minor, but they allowed us to

consider the modified SkNN algorithm in our evaluation. The more secure protocol hides the intermediate results from both the data owner and the service provider. Instead of processing each step on a single server, the data owner and the service provider work together to encrypt values and evaluate comparisons in a bitwise manner, without divulging plaintext to either server in the process. The iterative process results in the service provider having a list of encrypted records that correspond to the set of AOIs that are closer to a user's location than each AOI's threshold. SkNN operates directly on the user location coordinates and the coordinates of AOIs. Each AOI includes a threshold, which defines the boundary of the AOI, specified by a radius from the center of the AOI. This method does not require any location mapping to a grid system. The precision is determined by the number of decimal places specified by each location coordinates. SkNN uses the Paillier cryptosystem to support basic arithmetic operations to carry out primitive operations to determine closeness (or proximity).

The main difference between the two protocols is that the more secure protocol decomposes each location into individual bits to hide the true values from both servers (data owner and service provider.)

Privacy

Both protocols protect the confidentiality of the user's location from the data owner and the service provider (user location privacy). It also protects the database of AOIs from the user (DO privacy). The basic algorithm reveals query values to the service provider. When the service provider generates the top-k index and sends that back to the data owner, both the data owner and the service provider now know data access patterns. Our modifications to the basic SkNN protocol extend this step. In the new protocol, the service provider does not limit the index to top-k neighbors, but returns all AOIs that overlap user's location. The secure SkNN protocol protects the data

Algorithm 2 SkNN Basic Protocol (simplified)

1: The data owner encrypts a database of AOIs, (x_i, y_i), for $0 \leq i < m$, where m is the number of AOIs

2: The user encrypts location location, (x_a, y_a) then sends it, E(x_a, y_a), to the data owner

3: The data owner and the service provider calculate the Squared Squared Euclidean Distance (SSED) between the user's location and the center of each AOI, for $0 \leq i < m$. The data owner and the service provider call the Secure Multiplication (SM) protocol to calculate the squared values of each term, a and b_i, as follows (SM calculates the product of any two encrypted values. In our case, a = b):

(a) The data owner selects and encrypts 2 random numbers, r_a and r_b

(b) The data owner calculates a' = a * Enc(r_a), b' = b_i * Enc(r_b), then sends a' and b' to the service provider

(c) The service provider decrypts a' and b'

(d) The service provider calculates h' = Dec(a') * Dec(b') mod N, and sends h' to the data owner

(e) The data owner calculates the square of the supplied encrypted value:

i. s = h' * Enc(a)^(N-r_a)

ii. s' = s * Enc(b)^(N-r_b)

iii. Enc(a * b) = s' * Enc(r_a * r_b)^(N-1)

4: The data owner sends the encrypted squared distance vector to the service provider

5: The service provider decrypts the encrypted distance in each entry and creates and index list, corresponding to the points closer to the user's location than each AOI threshold

6: The service provider sends the index to the data owner

7: The data owner selects the records from the database of AOIs using the index and send the encrypted records to the service provider

8: The service provider decrypts the AOI records and send them to the user

access patterns (query privacy) by ensuring that the outputs of each step in the protocol are encrypted (only known to the data owner). Neither the data owner nor the service provider know which records belong to the current global minimum, and do not glean any information during the execution of the query. Permutation by the data owner prevents the service provider from associating tuples with the actual data, and the encrypted vector prevents the data owner from making associations with actual data. The secure protocol prevents the data owner and the service provider from knowing which data records correspond to the output set.

4.2.4 HCT: (Technique - Hilbert Curve)

Kim [42] offers a solution to protect the privacy of outsourced spatial data by proposing a method that uses a Hilbert Curve Transformation technique. The paper describes a method of creating a Hilbert Aggregation Index (HAI) and a Transformed Data Index (TDI) to increase the efficiency of query processing, while maintaining spatial data privacy. The proposed method minimizes the number and size of network messages between clients and server. The purpose of this method is to balance the speed requirements of range and kNN queries with privacy requirements to keep from disclosing actual location data to the service provider. The proposed scheme divides the entire spatial area into grid cells and assigns them cell IDs based on the Hilbert curve. HCT distributes trust and computation by defining the data owner as the trusted component, the service provider as the untrusted component, and the user as the computation component. Algorithms 3 and 4 list the steps in HCT.

Algorithm 3 HCT Outsourcing Spatial Data Protocol

1: The data owner generates TDI and HAI from the AOIs. The data owner retrieves fan-out (F) AOI cells. If data item count in a cell is less than F, the search expands the area along the Hilbert curve. When the data owner finds F AOI cells, the data owner stores IDs and location data in TDI and HAI IDs of the first and last grid cells of the search area. The data owner repeats until all AOI cells are encoded.
2: The data owner encrypts TDI and HAI entries with AES and sends to the service provider
3: The data owner sends the service provider a service token (to validate HAI version)
4: The data owner sends users the key for decrypting the data

Privacy

HCT provides the highest level of user location privacy of the three methods we implemented. Using HCT, the user never sends location (even in encrypted format) to any other entity. The cost of this protection is that the user must request AOI information from the service provider and carry out all computations locally. DO privacy is provided, within certain parameters. The data owner creates the TDI (a list of AOI cells, each entry having "fan out" (F) points), and then the HAI (an index containing starting and ending cell ranges for TDI entries), as an abstraction of the granular AOI location data expressed as distances on a Hilbert Curve. Since the data owner constructs TDI using the "fan out" value, any entity with HAI could only guess the actual cell of an AOI with a 1 / F probability of success. This is analogous to implementing k-anonymity over AOI locations. Thus, the data owner's AOI data is said to have the k-anonymity property, where k is equal to the F value chosen when building TDI structures. Since user has the decryption key, user can decrypt HAI and derive some information about

Algorithm 4 HCT Range query Protocol

1: User issues a query with a service token. If the service token of the User is not identical to that of the service provider, the user performs the HAI synchronization phase.

i. In this phase, the user requests a service token from the data owner. The user requests the encrypted HAI from the service provider by sending the service token received from the data owner.

ii. After the confirming the service token, the service provider sends the encrypted HAI to the user.

iii. Using the transformation key sent from the data owner, the user decrypts the encrypted HAI. Because the HAI synchronization phase is performed only when the service token is changed, the communication cost for the query processing can be greatly reduced.

Next, using the Hilbert curve, the user searches the IDs of grid cells that overlap a query region and retrieves the HAI records corresponding to the grid cells.

2: When the user searches the HAI records that satisfy the range query, the user sends the IDs of HAI records to the service provider.

3: The service provider sends the TDI records corresponding to the received IDs to the user.

4: The user decrypts TDI records using a decryption key and filters data items that are not located inside the query region. As a result, the user obtains the final result set of the range query.

the AOI locations. The final privacy property, query privacy, is partially provided by HCT protocol. The service provider can record TDI records the user requests and can associate general locations with TDI records. TDI entries are encrypted with AES, and the service provider does not possess the decryption key, but IDs of TDI records are not hidden from the service provider. The service provider could use query history to guess when any user is close to some other user from some time in the past, but cannot associate that information with either specific AOIs or user locations. It is possible that determining one user's proximity to another user in the past may divulge enough information to make other assumptions. Thus, the query privacy property is weakly protected in HCT.

4.2.5 Other Methods - Not Implemented

HVE: (Technique - Searchable Encryption)

Ghinita [30] proposes a privacy-preserving approach to providing alerts to subscribed users based on their location. Example applications include sending alerts to users in proximity to some emergency or restricted zone. The method in this paper is based on using searchable encryption techniques to allow an untrusted server to determine proximity to alert zones without knowing any user's actual location. We chose to not implement and assess HVE as it represents a method of encoding locations and is somewhat similar to concept to SBF, although the implementation differs substantially. We do revisit HVE and implement it for use in our extended framework, described in Chapter 5.

The method uses Hidden Vector Encryption (HVE) to provide searchability across encrypted data, supporting exact matching, range and sub-range queries, all on encrypted data. HVE uses bilinear maps and carries a high performance cost. To provide usable scalability, the authors propose op-

timized representation of alert zones (using a hierarchical representation), and the storing of often reused mathematical operations in the proximity detection process.

To receive alerts, Alice first contacts a Trusted Authority, Karen, to subscribe to alerts and receives a public key (PK) for the subscribed alert types. Alice then uses PK to periodically encrypt her current location and send the ciphertext to the untrusted server, Bob. When the Karen needs to define a new alert zone, it creates a search token, using a hierarchical representation of the cells contained by the alert zone. This search token is encrypted and sent to Bob. Bob evaluates the search token and current subscriber locations to determine if any user locations intersect the alert zone. Bob can send a notification to the affected user, or notify Karen of the intersection.

This approach has the advantage of focusing on the problem of determining proximity of one or more users and some alert zone, without revealing any user's actual location to Bob. The only information Bob learns is whether any user intersects with an alert zone at a specific point in time. Karen does have the ability to decrypt user locations, since it owns SK, but since Karen is trusted, this does not reduce the security guarantees.

Near-Pri: (Technique - Homomorphic Encryption)

Near-Pri [55] is an implementation of a scheme to provide private proximity based location sharing. The complete protocol allows users to share their specific locations once proximity (within a specified threshold) is determined. A slightly simplified version of the protocol could terminate once proximity is determined. (Thereby protecting the true location of each user and only reporting whether users are in proximity to one another.) We chose to not implement Near-Pri as part of our evaluation since it is essentially a peer-to-peer technique, as opposed to a user proximity to an authority-defined AOI technique.

The Near-Pri design in the paper uses Facebook chat to provide massage passing support and friend association. A user, Alice, sends a location query to another user, Bob. Bob builds a tree segment, a compact representation (in tree form) of his location. This location representation is critical to the efficient operation of Near-Pri. Bob creates two trees, one for longitude and one for latitude. The leaf nodes are numbered and organized so that locations surrounding any point can easily be isolated. Bob then determines his wall set (consisting of points below an arc that represents locations within Bob's policy threshold), and constructs n polynomials, based on each value of his wall set, encrypts the negated value of each coefficient, and sends these encrypted values and the public key used in the encryption to Alice.

Alice can evaluate each coefficient, encrypts the evaluation results, and sends them back to Bob. Bob decrypts each value, and if any value decrypts to zero. Bob knows that his wall set intersects with Alice's path set (i.e. Alice and Bob are within Bob's proximity policy.)

The Near-Pri protocol then requires Bob to reuse Alice's public key, which he uses to encrypt his own location. Bob sends his encrypted location to Alice, who can then decrypt it (using her private key), and then determine Bob's location.

4.3 Privacy Comparison

The privacy guarantees, as discussed earlier in each protocol section, differ as well. Table 4.2 summarizes the privacy guarantees of each method we implemented. If user location privacy is of the utmost concern, HCT never sends a user's location to any other entity, and therefore provides the highest level of user location privacy, followed by SkNN. SkNN does have the user send current location to the data owner, but the protocol carefully ensures that the data owner cannot decrypt the user location and the service provider

Table 4.2: Comparison of privacy guarantees

Method	**User location privacy**	**DO privacy**	**Query privacy**
SBF	k-anonymity based on filter size. The data owner learns when user is in an AOI	User only knows if location overlaps AOI	The data owner only sees obfuscated results - cannot correlate query to user
SkNN	The data owner only learns when user location overlaps an AOI	User only knows if location overlaps AOI	Extended protocol: The data owner/service provider cannot correlate queries to users
HCT	User location is never shared with the service provider, the data owner, or any other user	k-anonymity guarantees based on fan-out (F) value	Limited, the service provider learns proximity of user to AOI with $1/F$ accuracy

never has enough information to associate decrypted information with a user (or any AOI.) SBF also protects user location privacy by obfuscating the SBF representing the user's location when it is used for computation by the service provider. And finally, each protocol provides differing levels of privacy guarantees for queries issued. SkNN is designed specifically to limit any data leakage, including query information. Neither the data owner nor the service provider ever have enough information to know which queries are associated with any individual user. SBF provides a high level of query privacy by ensuring that the data owner only sees obfuscated computation results (non-zero SBF entries), but never enough information to determine the user's location. And as discussed above, HCT protocol does divulge some query history, but does not allow the service provider to determine any user's actual location.

4.4 Experiments

We implemented each algorithm and ran multiple tests to evaluate performance as input data size varied. We ran each test by first defining a set of AOIs, ranging from 1 to 100, then issuing 1,000 location proximity queries, each with a random user location. The times reported in the results represent the average time to resolve a single location proximity query. Our test engine recorded the result of each query, along with the "true" return value to assess each algorithm's accuracy. Accuracy results are expressed as the percentage of correct results returned over 1,000 queries. All tests were run on a single computer with 16GB memory and an Intel Core i7 2.4GHz CPU running Ubuntu Linux 15.10. We chose to run all tests on a single computer to focus on computation load. In future work we will expand the evaluation to measure communication overhead. Tests were run for varying numbers of AOIs, ranging from 1 to 100, using grid sizes from 10x10, up to 1000x1000,

and AOI sizes ranging from 1 to 9 units (or 9 to 361 cells).

4.4.1 SkNN

We implemented the Secure k-Nearest Neighbor algorithm in the C language. We also extended the paper's original functionality to support distinct location queries. The paper defined the kNN query in a classic sense, that is, to determine the nearest neighbors to a set of attributes by calculating the Euclidean distance between two vectors. Since our problem domain focuses on locations, we modified the Secure Squared Euclidean Distance (SSED) function to calculate Euclidean distance between two points. We also included a distance threshold with each AOI coordinate. The resulting algorithm only returns the AOIs that are closer to the user location than the AOI distance threshold. In this way, we allow the service provider to define AOIs by designating the AOI center point and the radius of the AOI. One drawback to this method is that AOIs can only be defined as circular regions. However, multiple circular AOIs can be defined to represent irregular AOI groups as necessary. The SkNN algorithm successfully determined proximity 100% of the time. This is expected since the SkNN algorithm compares distinct encrypted locations to determine proximity with the same precision as comparing unencrypted locations.

4.4.2 HCT

We implemented the Hilbert Curve Transition algorithm [42] in the C++ language. The HCT algorithm successfully determined proximity 94% of the time. Since the HCT algorithm maps true locations to grid cells, there is a small loss of accuracy. Our tests show that accuracy loss is tolerable, but non-trivial.

Table 4.3: Factors affecting performance

Method	**Num AOIs**	**AOI Size**	**Grid Size**
Spatial Bloom Filter (SBF)	No	No	Yes
Secure k-Nearest Neighbor (SkNN)	Yes	No	No
Hilbert Curve Transformation (HCT)	Yes	Yes	Yes

4.4.3 SBF

We implemented the Spatial Bloom Filter algorithm for location privacy [11, 56] in the C language. The SBF algorithm successfully determined proximity over 99% of the time. Although Bloom Filters can exhibit false positives, in our experiments, we only observed false positives in less than 1% of our tests.

4.4.4 Input variable impact on performance

Each algorithm showed different scalability patterns as the number of AOIs, size of AOIs, and size of the grid increased. Comparing the raw run times of the implemented algorithms is of less interest than examining how each reacts to input domain data size growth. Any of the algorithms can be optimized to increase overall performance prior to release in a production environment. The results in Figure 4.3, Figure 4.4, and Figure 4.5 represent the average runtime for a single query (i.e. user proximity query). The results only consider computation time and do not represent communication overhead. Table 4.3 summarizes the sensitivity to input domain data size growth.

Impact of AOI number

Figure 4.3 shows how changing the number of AOIs affects performance. Test were run with constant AOI and grid sizes, and the number of AOIs ranging

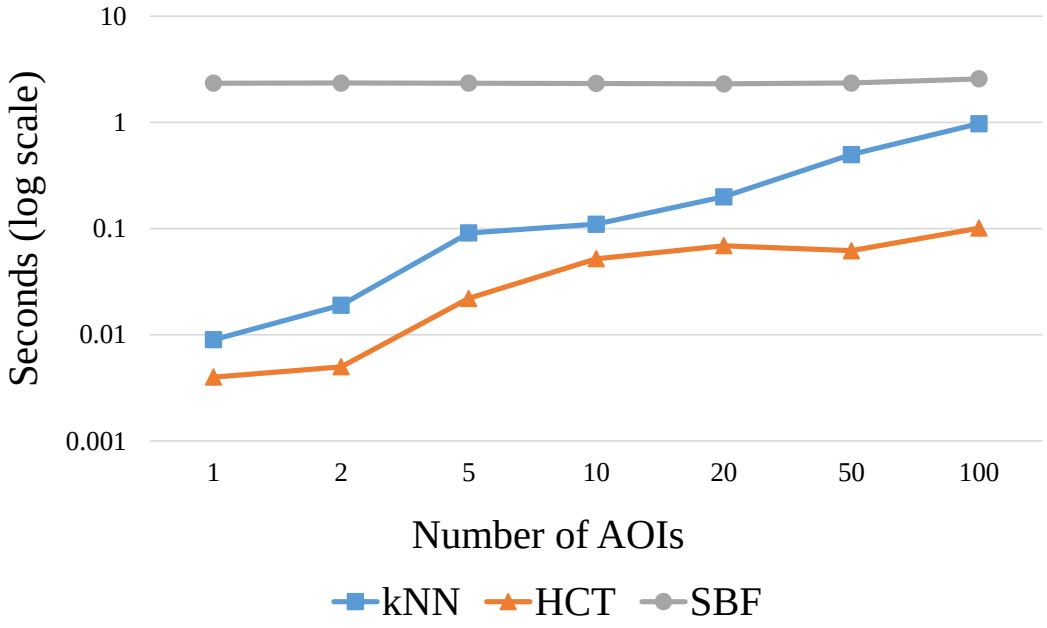


Figure 4.3: AOI number impact on performance

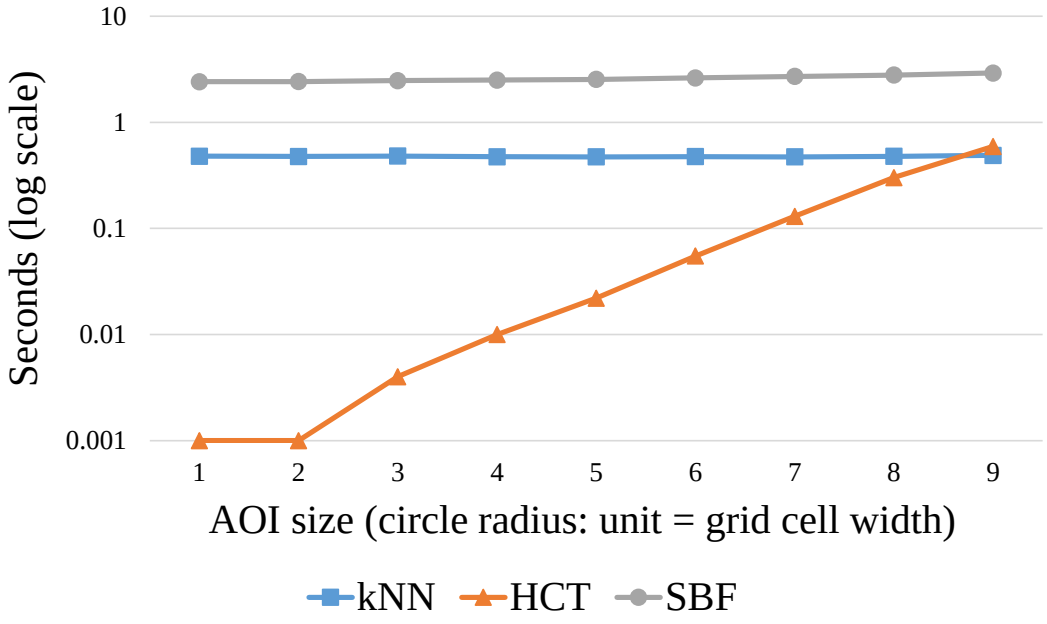


Figure 4.4: AOI size size impact on performance

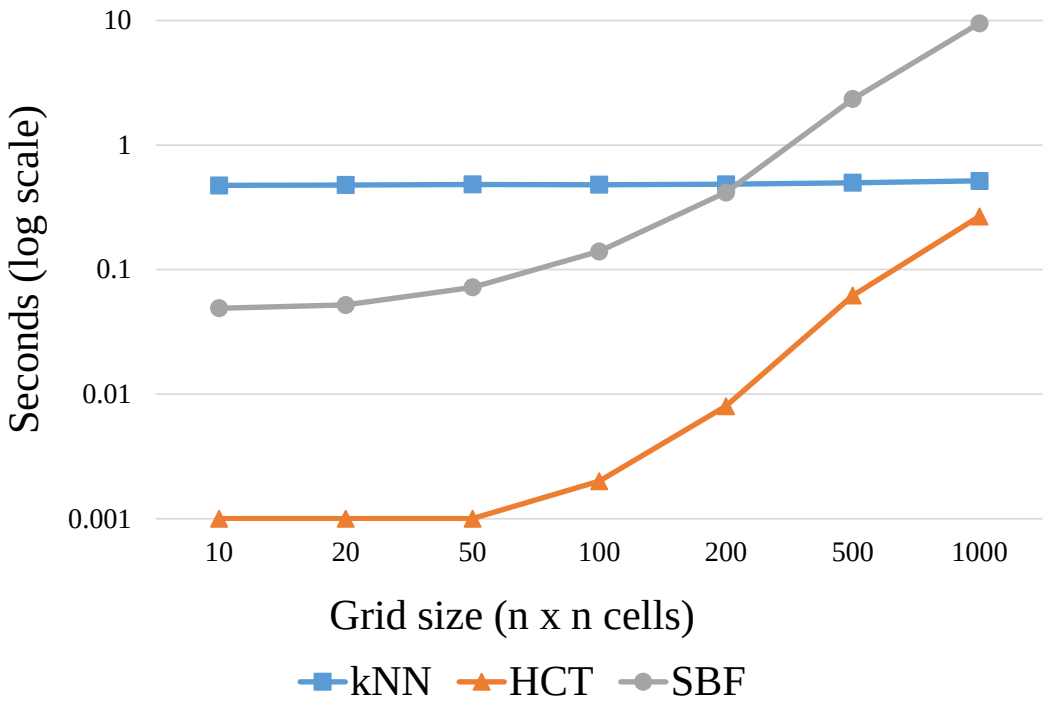


Figure 4.5: Grid size impact on performance

from 1 to 100. SBF had the most stable performance in this test, but was also the slowest algorithm, although SkNN approached SBF run time when the number of AOIs exceeded 50. HCT run time increased with the number of AOIs, but was the fastest overall. It is expected that both SkNN and HCT would exhibit slower performance than SBF as the number of AOIs exceed 200.

Impact of AOI size

Figure 4.4 shows how changing the size of AOIs affects performance. We varied the size of AOIs from 1 to 9 units, keeping the number of AOIs and grid size constant. The size of each AOI depends on how locations and thresholds are defined. We simply use the term "unit", but real implementations will require a unit definition, such as each unit equals 1 mile or 1 km. HCT and SBF use cell groups to define AOIs. To standardize our comparison, we defined the cells in each AOI using the formula $aoiCells = ((r*2)+1)^2$, where r is the radius of the AOI. Neither SBF or SkNN algorithms were affected by AOI size. HCT's performance slowed as the number of AOIs increased.

Impact of Grid size

Figure 4.5 shows how changing the size of the grid affects performance. We varied the size of the grid from 10x10 to 1000x1000 cells, keeping the number and size of the AOIs constant. SkNN's performance was unaffected by grid size increase. HCT and SBF performance both degraded as the grid size grew. The performance degradation was nearly the same for HCT and SBF, and was most noticeable when the grid size grew larger than 50x50 cells.

4.4.5 Performance comparison

Based on our findings, each protocol satisfies different criteria in different ways. For environments that must cover large areas, SkNN was least affected by grid size. Since SkNN only evaluates defined AOIs, the actual grid size has essentially no effect on the protocol's performance. We did note that the raw performance numbers of HCT exceeded SkNN up to a grid size of 1000x1000. It is assumed that implementation optimizations could reduce the actual deployed run time of SkNN to compete with HCT at smaller grid sizes (<1000x1000). If the grid size is fairly constant, but the number of AOIs changes frequently, SBF provided the most consistent performance results. One side effect of SBF for very large grid sizes (>1000x1000) is that more hash functions are necessary to avoid collisions, which may negatively affect overall performance. And finally, if the primary variable is the AOI size, both SBF and SkNN are valid choices. Both of these protocols were unaffected by the size of AOIs. Although the greatest performance variance was measured with HCT, this protocol performed the best of all three for smaller AOI counts (<100), smaller AOI sizes (<7 units radius), and smaller grid sizes (<1000x1000). HCT uses smaller data structures with smaller overhead than the other protocols. Each of the methods we examined fit well into different use cases. Our findings support the following guidelines for choosing an method based purely on performance considerations:

- **SBF** - Best for small to medium grid size (<500x500), when number or size of AOIs vary
- **SkNN** - Best for environments with varying grid and AOI size, but with a limited number of AOIs (<50)
- **HCT** - Best for small to medium number (<100) and size of AOIs (<7 units radius), and small coverage area (<500x500)

4.5 Conclusions

We implemented the SBF and HCT algorithms as the authors presented, and extended the SkNN algorithm to support our problem statement. We evaluated the performance of the implemented algorithms and presented our results as the input domain of the number and size of AOIs, and size of the overall grid changed. We presented results comparisons, along with recommendations for choosing the best algorithm based on environmental requirements.

Our results can be used to select the best of our assessed techniques for specific environments, for example, Smart Cities. It is expected that Smart Cities will need to respond to the needs of its constituents based on growth and usage needs. For environments that experience radical increases and decreases in the number of AOIs it must manage, the SBF approach likely offers the most stable runtime performance. This recommendation is dependent on the assumption that the grid size of a Smart City is relatively small, as in smaller than 500x500 cells. If the selected cell size is 0.2km with a grid size of 500x500 cells, SBF would easily handle even the largest cities in the world. If a Smart City only needs to manage fewer than 100 AOIs that are relatively small (<7 units radius), then HCT may be a better solution. Using a grid size of 0.2km, that would mean that most AOIs would be <1km radius. Such a scenario could indicate that HCT performs better than SBF for a specific application. Regardless of the specific application parameters, our assessment provides tangible guidance for choosing the best approach for specific Smart City environments.

Chapter 5

Mutually Private Proxmity Detection in Categorical Settings

5.1 Problem Definition

This third work [64] focused on the logical next step indicated by the previous two works, that is, to extend MPPD to include users defined in different categories. In Chapter 4, we explored MPPD techniques, and additionally contributed a new technique to existing solutions. One drawback of all of the existing solutions is that they assume "all or nothing" access control decisions. The only way to define sets of AOIs to be consumed by a specific category of user would require defining separate AOI sets for each user category. This would increase administration and redundancy. In this chapter we present a solution that allows users defined in different categories, (i.e. that possess different attributes), to consume only authorized AOI location information while using a single set of AOIs.

5.1.1 Motivation

Consider Mary, who is a guest at the "Fun Times" amusement park. Mary wants to make the most of her day in the park and desires to minimize time spent waiting in line for attractions or shows. Mary has subscribed to the "Fun Times InTheKnow app" premium service that sends information to her smart phone about nearby attractions and shows with short wait times. Bob is also in the "Fun Times" park and has the "InTheKnow" app, but Bob did not subscribe to the premium service. Bob only receives general information about attraction wait times for attractions that are somewhat close to his current location. Both Mary and Bob want to receive helpful information without disclosing their locations to "Fun Times". On the other hand, "Fun Times" wants to provide location-sensitive services to Mary and Bob without publishing all of the "Fun Times" areas of interest. "Fun Times" can limit the information they provide to subscribers, and even define different subscriber levels based on subscription fees paid. "Fun Times" also uses this service to direct their employees to areas of the park that need cleaning, servicing, or even crowd management. Other use cases for such a location privacy preserving framework could include providing epidemiologic related alerts or criminal activity investigation proximity with precision granularity based on clearance and/or "need to know".

5.2 Problem Statement

Existing MPPD solutions largely focus on location as the only criteria for determining proximity to a defined area. In some cases, additional attributes should be considered, such as security level or subscriber status, before providing proximity responses. A useful MPPD framework should allow a single set of AOIs to service a large group of diverse users. Such a framework would

only provide proximity alerts for users that are both near a defined AOI and meet requirements set by the data owner. This is normally accomplished by separating the MPPD functionality from the authorization phase. This requires a third party to examine each user and AOI attributes to determine of a proximity alert is allowed, based on data owner policy. In this scenario, the third party possesses substantial information about users and AOIs. Our framework provides fine-grained access control that is embedded in the encryption technique. That is, decryption is only successful when attempted with an authorized user's key. Users are authorized to decrypt AOI information based on their attributes and the policy provided my the data owner. This feature allows the data owner to control which types of users can decrypt (and access) their AOI data without knowing which users will access the data in advance. In addition to our use of CP-ABE for fine-grained access control, our framework uses HVE to determine proximity without disclosing any location data to a third party. The HVE decryption succeeds only when the user's location (i.e. the one who submits the proximity query) is near an AOI. The service provider learns nothing more than whether a user is near one or more AOIs, or not. And since HVE supports compressed tokens, (i.e. a single token uses wildcards to represent multiple cell locations), the proximity detection computational load is reduced. For example, Figure 5.1 shows two applications and three types of users. In the first application, a general user asks "Am I near a dangerous area?" A firetruck driver asks the same question. The answer may be different, and the action taken by each party should be different. The general user should receive a general warning and should stay clear of the dangerous area, while the firetruck driver should be given precise location information for the emergency. In the second application, an amusement park visitor is looking for short lines. The amusement park application has that information, but will reply with different types on information depending on the subscription status of the user.

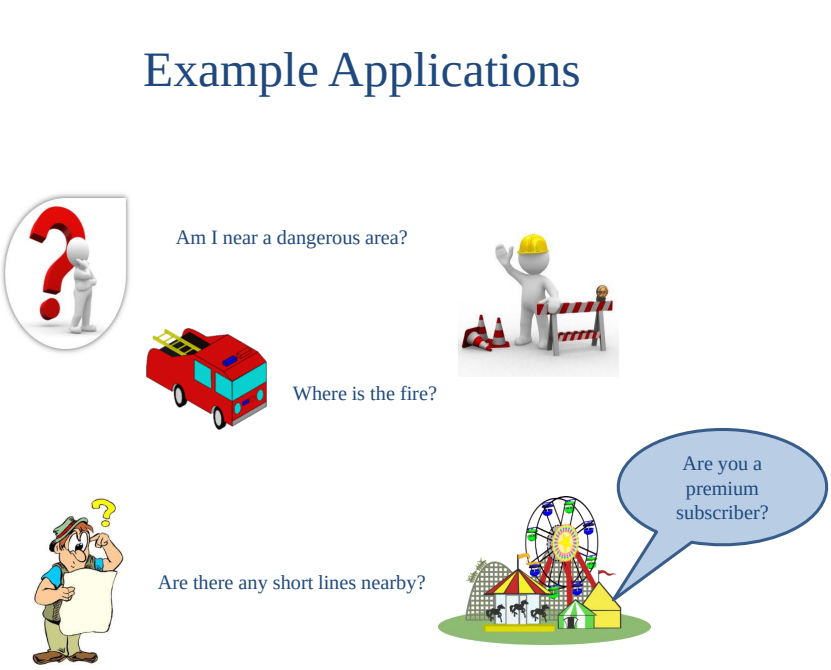


Figure 5.1: CP-ABE Access Policy Tree

5.3 Background

5.3.1 Framework Model

Our location-based proximity detection model is based on users traveling through real space represented by a two-dimensional grid. At any one point in time, a user occupies exactly one grid cell. On demand, users can query an encrypted database of AOIs, using their own encrypted locations, to determine if their current location overlaps any AOI. The data owner defines multiple AOIs, encrypts them, and supplied them to the service provider. The service provider carries out the calculations on encrypted data to determine if the user's current location is in proximity to one or more AOIs. The service provider responds to the user with a list of AOIs that overlap the cell that encloses the user's location. Cells can represent any physical size. The only restriction is that the data owner and all users must use the same cell granularity when converting AOI or user locations into grid coordinates.

Our protocol uses an architecture of four distinct entities. They are:

- Data Owner/Provider (DO) - Defines/encrypts AOI locations/access policies
- Key Generator (KG) - Generates CP-ABE and HVE keys and tokens.
- Service provider (SP) - Provides computation services for users to determine user proximity to any AOI
- User - user with ability to determine current location (GPS or other means)

The function of each entity at each step of the protocol is explained in a later section.

5.3.2 Privacy Model

An important privacy requirement of an MPPD scheme is that the data owner learns nothing about user locations and users learn nothing about AOI locations (defined by the data owner), except in the case when their current location overlaps one or more AOIs. Even when a user learns that he or she overlaps an AOI, the complete dimensions of the AOI are undisclosed. A user could travel around the grid in a structured attempt to learn AOI locations by remembering cells with AOI overlaps. Our framework and protocol does not protect eventual AOI location disclosure from such an attack. Protocol extensions to protect AOI locations from deliberate user brute-force attacks are left for future work, but could be implemented at the key generator by having the key generator detect patterns of such attacks and responding to the user accordingly. The key generator does know the location of a user when that user requests an HVE key, but the key generator is trusted, and thus does not share user location with any other entity and does not have access to any AOI information. The key generator does not return the generated HVE encrypted user location to the user. Instead the key generator sends the encrypted user to the service provider. The service provider only has access to encrypted AOI and user locations. It can only learn that a user's location overlaps one or more AOIs, but does not know what location the overlap represents in the gird. The service provider could infer user to user proximity when multiple users are in proximity to the same AOI within a short period of time. But the service provider still would not know the location of any user or AOI. Although HVE is a public key cryptography scheme and would provide the possibility for a malicious service provider to attempt to build a fake encrypted AOI list, our framework limits the distribution of the HVE public keys. This practice limits the ability to encrypt AOI locations to only trusted entities, thereby removing the ability to use dictionary type attacks to generate imposter AOI lists.

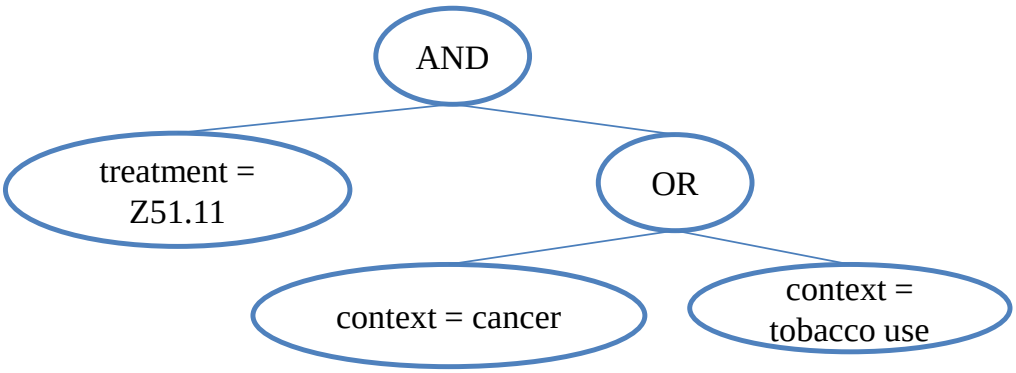


Figure 5.2: CP-ABE Access Policy Tree

5.3.3 Ciphertext Policy Attribute Based Encryption

Recall from Chapter 3, a CP-ABE scheme provides fine-grained access control over data [4]. CP-ABE associates a user with a set of descriptive attributes to generate the user's secret key, SK. Data are encrypted under an access policy such that only users whose attributes match the access policy can decrypt the data. To encrypt a message M using CP-ABE, the encryptor provides an access policy which is expressed as a boolean expression containing selected attributes and values for M. Figure 5.2 shows the access policy presented earlier in a tree structure. The message is then encrypted based on the access structure, T. Decryptors generate SK based on their attributes. A decryptor is only able to decrypt ciphertext, CT, when her SK satisfies the access policy used to encrypt the message. Unauthorized users cannot decrypt CT even if they collude and combine their disjoint attributes.

CP-ABE defines the following four essential functions:

1. Setup(): Input security parameter, output public key (PK), for encryption, and master key (MK), to generate user secret keys.

2. Encrypt: Input message M, access structure T, public key PK, output ciphertext CT.

3. KenGen: Input set of user's attributes SX and MK, output secret key SK for SX.

4. Decrypt: Input CT, SK. If SK satisfies access structure in CT, return M, else return NULL.

5.3.4 Hidden Vector Encryption

Hidden Vector Encryption (HVE) [8] is an extension of an anonymous identity based encryption (IBE) [6] scheme. With IBE, the keys used for encryption and decryption are based on identities and attributes. HVE allows an attribute string that is associated with the ciphertext or the user secret key to contain wildcards. Thus, HVE provides a searchable encryption scheme that supports conjunctive equality, range and subset queries. An HVE attribute is represented as a vector of elements with a value of 0, 1, or a wildcard (represented as '*' and often referred to as a "don't care" value). The wildcard in an HVE attribute matches the values 0 or 1 in comparison operations. An HVE comparison of a search predicate S and a ciphertext C evaluates as True if the attribute vector I used to encrypt C contains the same values as S for all positions that are not '*'.

HVE defines the following four essential functions:

1. Setup(): Input security parameter, output public key (PK), for encryption, and master key (MK).

2. KeyGen: Input MK and a string y in 0, 1, *, output secret key SK for y.

3. Encrypt: Input public key PK, message M, attribute string x in 0, 1, output ciphertext CT.

4. Query: Input CT, SK. If SK satisfies attribute string x in CT, return M, else return NULL.

Note that the HVE setup() function returns PK and MK. In most public key encryption schemes, PK is intended for public consumption. Any user can access PK and can generate ciphertext. In our framework, we want to limit the ability for create ciphertext, (encrypted AOIs), to trusted entities, specifically, data owner. Of course, since the key generator generates PK, it has access to PK as well. Thus, each data owner protects the privacy of its own AOIs by protecting (i.e. not disclosing) its own PK for HVE encryption. While it is true that any data owner could collude with a service provider, that act would violate the data owner's stated goal of protecting the privacy of its own AOI locations.

5.4 Protocol Description

Suppose the "Fun Times" park defines four types of AOIs, each with a different color designator. Table 5.1 lists the AOI types and what each one represents. These AOI types are simply examples of what our proposed framework can represent. AOI types can be of any type and any number. The AOI type definition is left up to the specific implementation definition. The only requirement is that each AOI must be uniquely identified with some character label. AOIs can overlap one another by sharing one or more defined cells. In such cases, users would receive a response to a proximity query indicating that their current location places them within all AOIs that contain their current cell. Using our example, "Fun Times" wants to provide some value to users who use their mobile app, but reserve the more detailed information for premium subscribers to their service. Users of the "InTheKnow" app that have paid for the premium service can receive "warn", "notify", and "approach" alerts. The first two alert types could be used to provide

Table 5.1: AOI Types

Color	**Who can access**	**Alert type**
Red	Paid subscribers	Warn
Blue	Paid subscribers	Notify
Green	Paid subscribers	Approach
Yellow	Free users	Notify

guidance for areas to avoid, while the third alert type could inform users of areas that would be beneficial to visit. Users that have not paid for the premium subscription will only receive "notify" alerts. Notice that there are two different colors for the "notify" alert. Premium subscribers will receive more specific information about "notify" alerts, while free users will only receive general messages.

Figure 5.3 shows how each AOI accessible by paid subscribers is defined on a grid and the access policy associated with each AOI. Figure 5.4 shows the AOI accessible by free users and its associated access policy.

Our example includes four users to demonstrate the protocol's flexibility. Figure 5.5 shows each user and their associated descriptive attributes used to generate each user's secret key. A user's attributes must satisfy (match) an AOI's access policy to decrypt and access the AOI.

To determine proximity to an AOI, a user sends the current encrypted location to a service provider, which then determines if that user is within any cell defined as an AOI, all without ever learning any location information from the user or data owner. Users request a secret key from a key generator based on descriptive attributes. The attribute-based key provides the ability for the service provider to assess accessibility for each AOI. To continue our example, four users request proximity alerts for the amusement park defined AOIs.

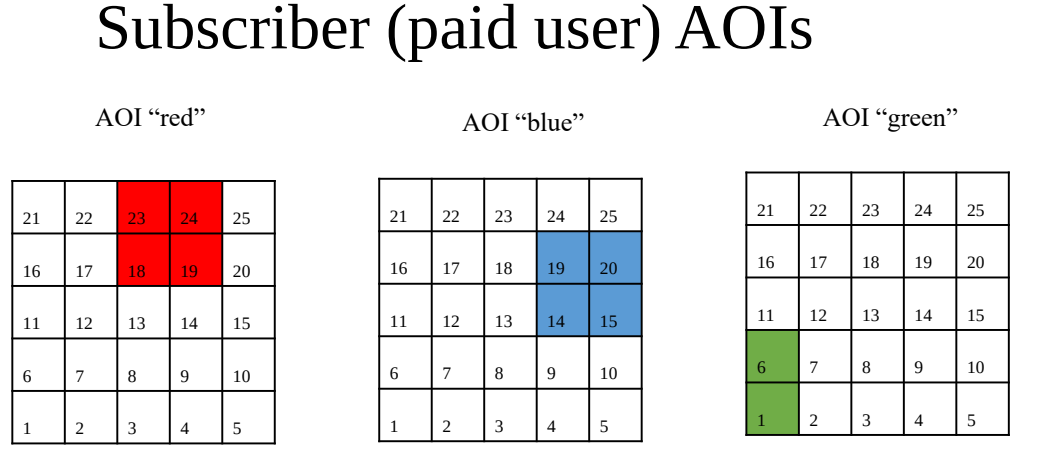
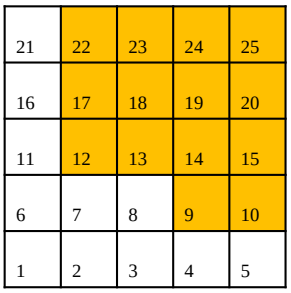


Figure 5.3: Subscriber (paid user) AOIs

Non-subscriber (free user) AOIs

AOI “yellow”



AOIs for free users are more generic, less granular

(i.e. provide less specific information to free users)

Access policy: AOI “yellow”

“(subscriber=free) AND
(alertType=notify)”

Figure 5.4: Non-subscriber (free user) AOIs

User access to AOIs

User	Subscriber	AOI types	Can decrypt AIOs
Alice	free	warn, notify	yellow
Bart	free	approach	None
Mary	paid	approach, notify	Blue, green
Daniel	paid	warn, notify	Red, blue

AFTER decrypting, we can determine user proximity to AOI cell

Figure 5.5: Users and Descriptive Attributes

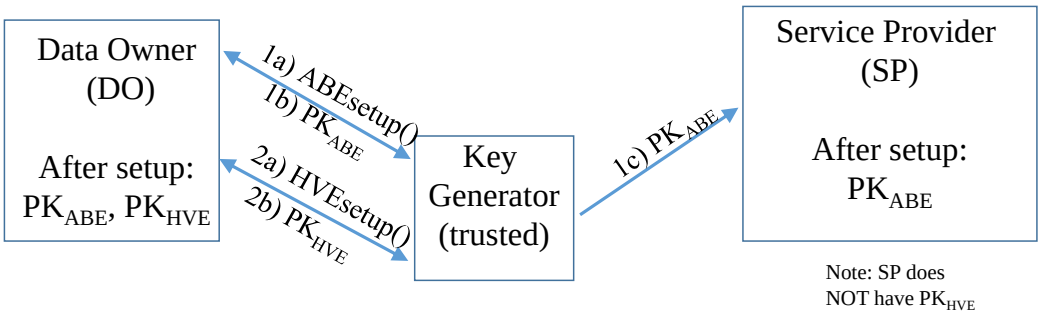


Figure 5.6: PrivProxABE Setup phase

We call the technique PrivProxABE (MMPD using ABE) [64]. The protocol is made up of four basic phases:

I Setup - Initializes protocol state

II InitAOIs - Encrypts AOIs with access policy

III InitUserLoc - Encrypts current user location

IV Query - User-initiated location proximity query

5.4.1 Setup

Algorithm 1 shows the steps in the "Setup" phase and refers to figure 5.6. In the "Setup" phase, the data owner calls the ABEsetup() method on the key generator to generate the ABE public key, PK_{ABE}. The key generator sends PK_{ABE} to the data owner and the service provider. The data owner also calls the HVEsetup() function on the key generator to generate the HVE public key, PK_{HVE}, and returns PK_{HVE} to the data owner. The data owner keeps PK_{ABE} and PK_{HVE} secret, (i.e. the data owner does not share either key with any other entity.)

Algorithm 5 PrivProxABE Protocol - Setup

1. The data owner calls ABEsetup() on key generator. The key generator sends PK_{ABE} to the data owner and service provider.
2. The data owner calls HVEsetup() on key generator. The key generator sends PK_{HVE} to the data owner.

5.4.2 InitAOIs

Algorithm 2 shows the steps in the "InitAOIs" phase and refers to Figure 5.9.

Encode AOI Locations

In the "InitAOIs" phase, the data owner first maps cell IDs to coordinates, (x,y), and the generates Gray codes for each point in the ordered pair. The complete Gray code for a cell ID is the concatenation of the ordered pair's x value and the ordered pair's y value. Figure 5.7 shows the Gray codes for the "red", "blue", "green", and "yellow" AOIs presented in Figure 5.3 and Figure 5.4 .

To reduce the number of comparisons necessary to determine AOI proximity during user queries, we compress the Gray codes into search tokens that contain wildcards, as proposed in [30] . If two Gray code values differ by only a single digit, we replace the digit with a wildcard, "*", and thus a single search token can represent two individual Gray codes, or Cell IDs. We continue the compression process iteratively until no two remaining search tokens differ by only a single digit. This process can compress multiple Cell IDs into a small number of search tokens.

For example, the Gray codes for cell 18 (0001000110) and cell 19 (0011000110) only differ by the value in position 2. Therefore, we can combine the two Gray codes into a single token, replacing the value in position 2 with a wildcard "*", (00*1000110). The wildcard represents a "don't care" value, since

AOI Encoding – Step 1

AOI color	Cell ID	Cell Coordinate	Gray code
red	18	(3,3)	0001000110
red	19	(4,3)	0011000110
red	23	(3,4)	0001000111
red	24	(4,4)	0011000111
blue	14	(4,3)	0011000010
blue	15	(5,3)	0011100010
blue	19	(4,4)	0011000110
blue	20	(5,4)	0011100110
green	1	(1,1)	0000100001
green	6	(1,1)	0000100011
yellow	9	(4,2)	0011000011
yellow	10	(5,2)	0011100011
yellow	12	(2,3)	0001100010
yellow	13	(3,3)	0001000010
yellow	14	(4,3)	0011000010
yellow	15	(5,3)	0011100010
yellow	17	(2,4)	0001100110
yellow	18	(3,4)	0001000110
yellow	19	(4,4)	0011000110
yellow	20	(5,4)	0011100110
yellow	22	(2,5)	0001100111
yellow	23	(3,5)	0001000111
yellow	24	(4,5)	0011000111
yellow	25	(5,5)	0011100111

Figure 5.7: AOI Location Encoding - Step 1

the token matches any value in position 2. The Gray codes for cell 23 and 24 also differ by only a single digit and can be represented with the token (00*1000111). Notice that the two resulting tokens differ by only a single digit and can also be combined into a single token (00*100011*). In this way, a single token can represent four cells. Figure 5.8 shows the result of the compression process for the “red”, “blue”, “green”, and “yellow” AOIs presented in Figure 5.3 and Figure 5.4 .

Encrypt Encoded AOIs

After compressing each of the Gray encoded AOI locations, each AOI is represented by one or more compressed tokens. The data owner encrypts each compressed token with HVE using PK_{HVE}, returning CT_{HVE}. The data owner then encrypts the AOI label and any additional AOI information for a single AOI with CP-ABE using PK_{ABE}, returning CT_{ABE}. The data owner

AOI Encoding – Step 2

AOI color	Compressed Gray code token(s)
red	00*100011*
blue	0011*00*10
green	00001000*1
yellow	0011*00*1*, 0001*00*10, 0001*00111

Figure 5.8: AOI Location Encoding - Step 2

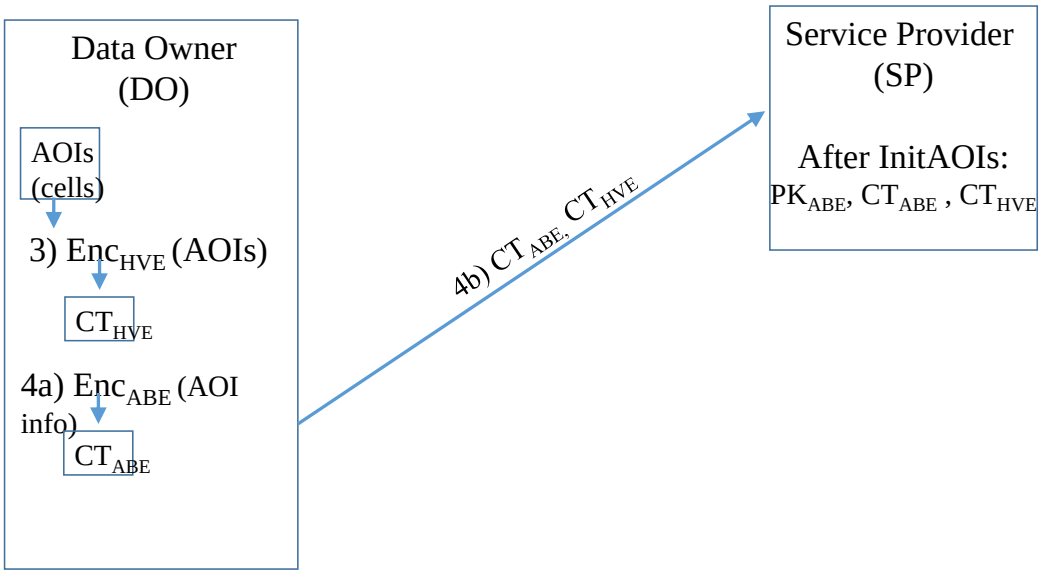


Figure 5.9: PrivProxABE InitAOIs phase

then sends CT_{ABE} and CT_{HVE} for all AOIs to the service provider.

Algorithm 6 PrivProxABE Protocol - InitAOIs

3. The data owner encrypts encoded AOI cells (encoded using Gray encoding) and keeps CT_{HVE}.

4a. The data owner encrypts AOI label and other descriptive AOI info using access policy to get CT_{ABE}.

4b. The data owner sends CT_{ABE} and CT_{HVE} to the service provider.

5.4.3 InitUserLoc

Algorithm 3 shows the steps in the "InitUserLoc" phase and refers to Figure 5.10. If the user is new (e.g. not a user known to the key generator), the key generator authenticates the user and records the authorized user attributes in the key generator user table. The key generator uses the user attributes to generate a secret key (SK_{ABE}), and returns SK_{ABE} to the user. The user then calls HVEencrypt(currentLocation) on the key generator to encrypt the user's current location using HVE, generating Loc_{HVE} The key generator returns Loc_{HVE} to the user. The user then generates a randomIdentifier for use in the query phase. The randomIdentifier detaches queries from user identities.

Algorithm 7 PrivProxABE Protocol - InitUserLoc

5. User calls ABEkeyGen(userID) on key generator. The key generator authenticates new users and generates a CP-ABE secret key (SK_{ABE}) based on the user's attributes, and returns SK_{ABE} to the user.

6. User calls HVEencrypt(currentLocation) on key generator. The key generator returns the encrypted location (Loc_{HVE}) to the user. The user generates a randomIdentifier.

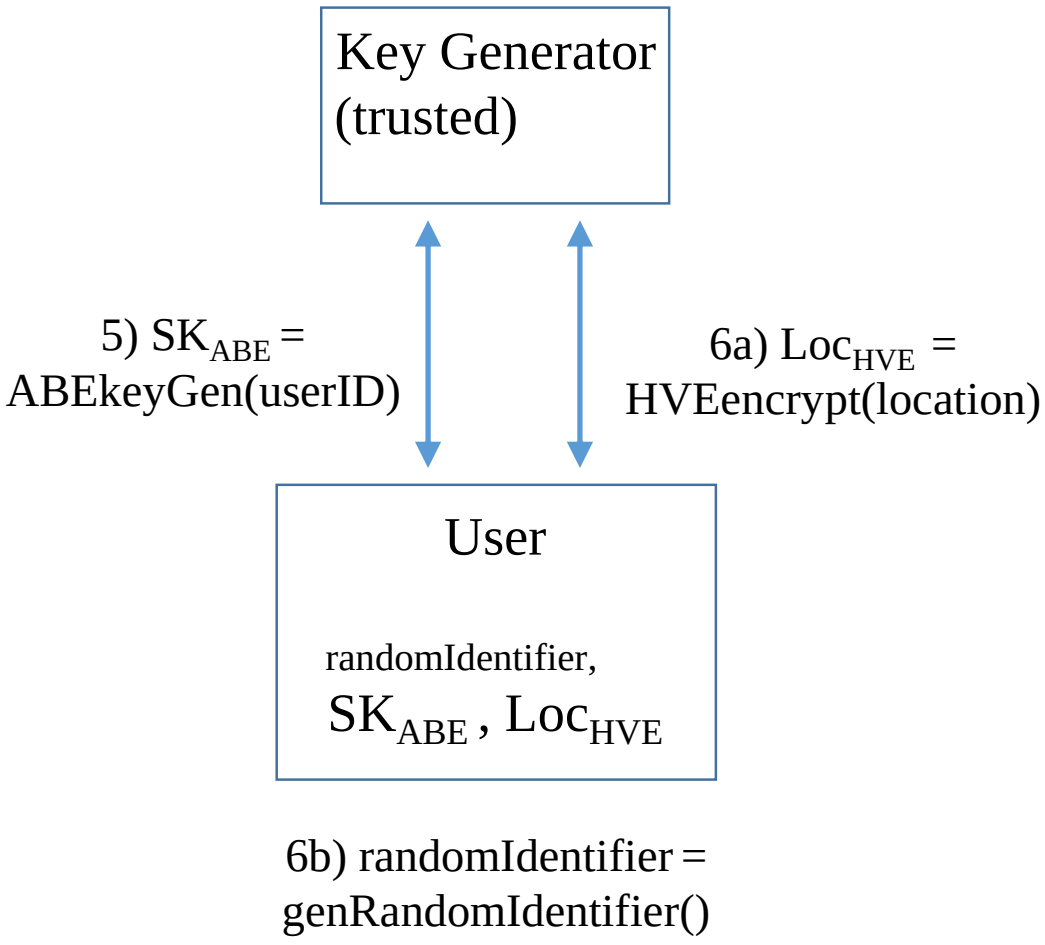


Figure 5.10: PrivProxABE InitUserLoc phase

5.4.4 Query

Algorithm 4 shows the steps in the "Query" phase and refers to Figure 5.11. The HVEdecrypt(Loc_{HVE}) function, run on the service provider, iterates through each AOI and attempts to decrypt CT_{HVE}. If HVE decryption is successful, that means that the user's location shares one cell defined by the AOI. The service provider returns a list of all successfully decrypted HVE ciphertexts to the user. The user runs ABEdecrypt(SK_{ABE},CT_{ABE}) to attempt to decrypt each AOI returned from the service provider. Successful ABE decryption means the user's attributes satisfy the AOI's access policy for an AOI that overlaps the current user's location. The service provider returns a list of encrypted AOIs that overlap the user's location. The user then attempts to decrypt each AOI using CP-ABE. Any successfully decrypted AOI means that the user's location overlaps the AOI and the user is authorized to consume the AOI's information.

Algorithm 8 PrivProxABE Protocol - Query

7. User calls HVEdecrypt() on the service provider. The service provider attempts to decrypt all AOIs using user's Loc_{HVE}.
8. The service provider returns a list of all successfully decrypted (HVE) AOIs.
9. User calls ABEdecrypt(SK_{ABE},CT_{ABE}) for each AOI returned from the service provider to determine if the user is authorized to consume proximity alerts for each AOI.

5.5 Security and Privacy

One of the primary requirements of this framework is to maintain the privacy of all actors. The proposed framework relies on the security guarantees of CP-ABE and HVE to provide the overall security of our approach. In each

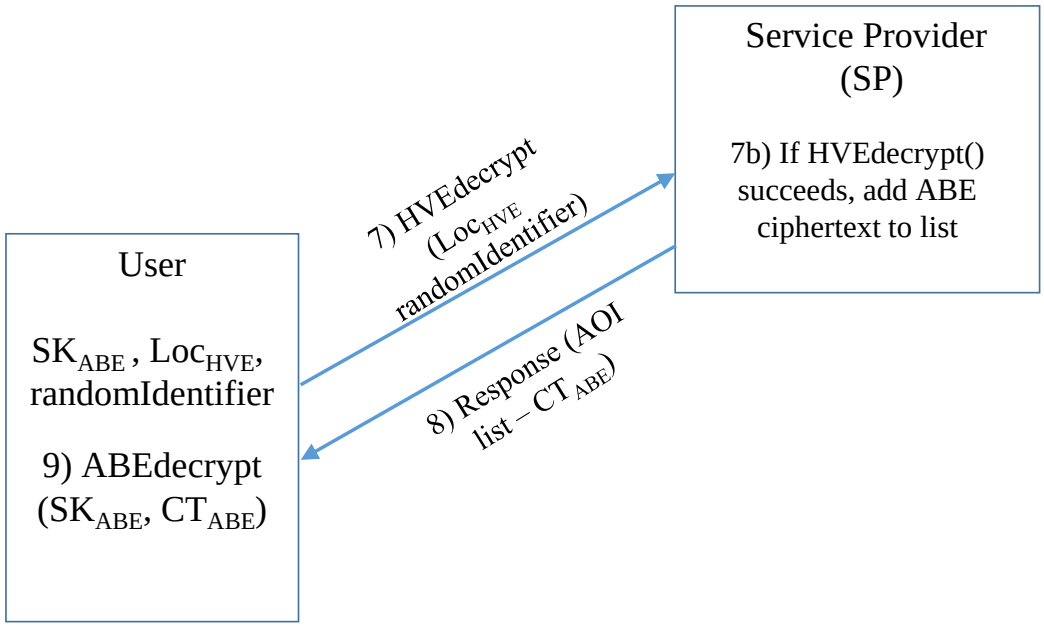


Figure 5.11: PrivProxABE Query phase

scheme, the SK is necessary to decrypt data. Our framework ensures that only trusted entities, users and the key generator, have access to any SK. Table 5.2 summarizes the privacy guarantees with respect to each architectural entity and the following list explains each of the privacy guarantees.

- Data Owner/Provider (DO) - The data owner is the only entity with access to the unencrypted AOI data. It does not share unencrypted AOI locations or any HVE keys with any other entity to protect AOI locations. The data owner does not have access to any other information generated or stored by the key generator or any user, and thus cannot learn any information about user locations.

- Key Generator (KG) - The key generator, a trusted entity, generate SKs, but never has access to AOIs and cannot decrypt any AOI information without colluding with another entity. The key generator does

Table 5.2: Factors affecting performance

Entity	**Privacy**
Data Owner/Provider (DO)	Does not share unencrypted AOI data or HVE keys. The DO has no access to other data.
Key Generator (KG)	Trusted entity with no access to AOI data. Does not share user location data with any other entity.
Service Provider (SP)	No access to unencrypted data. The SP can learn user/AOI proximity, but cannot correlate to real location.
User	User does not share location with any entity other than the KG. The user can spoof location to build a partial AOI map.

learn user locations when it generates Loc_{HVE} for a user's current location, but does not share this information with any entity other than the user and does not have access to any information, such as AOI locations, with which to correlate user locations. The key generator could remember user locations to build user trajectories, but our framework defines the key generator as a trusted entity and we expect that the key generator will not exceed its authority.

- Service provider (SP) - The service provider never has access to unencrypted AOI or user locations, and cannot access the decryption keys to decrypt any ciphertext. Thus, the service provider never learns any information about AOI or user locations. The service provider does learn that a user is in proximity to one or more AOIs when the locations overlap, but does not have any information to imply actual user or AOI locations. The service provider also can determine when two or more users are close to one another when they are found to be in proximity to the same AOI(s). The service provider cannot attempt a dictionary type attack by building a fake AOI list since it lacks PK_{HVE} which is necessary to encrypt AOI locations.

- User - The user only divulges actual location to the key generator. Since the service provider carries out all calculations on encrypted data, the service provider never learns the user's location, and as stated previously, the data owner never has access to any user location information. The user never has access to any AOI location and only learns general AOI information when the service provider discloses that the user is in proximity to one or more AOIs. The user learns that the current location overlaps the reported AOIs, but does not immediately learn the dimensions of any AOI. Through a process of strategically visiting multiple cells, a user could build a limited map of AOI locations based

on proximity alerts. However, a user could only construct meaningful map entries from AOI data the user is authorized to decrypt. One method to restrict malicious users from building even a partial AOI map in a short period of time could be to set a threshold of maximum speed at which a user could realistically travel from cell to cell. If a user attempts to exceed this threshold, he would be deemed malicious. Hallgree, et. al. [33] propose a protocol, MaxPace, based on the concept of using travel speed thresholds to detect malicious users. Implementing protection from this type of attack is left for future work.

Our protocol protects both AOI and user locations from disclosure, and only allows users to eventually build an AOI map after determined actions resulting in recording history of cells visited and proximity alerts.

5.6 Experiments

We implemented a prototype of our framework and Li's [51] similar framework and ran multiple tests to evaluate performance as input data size varied. We ran each test by first defining access policies, half of which are satisfied by test user attributes. So for each test, the user is authorized to query half of the AOIs. We then define a set of AOIs, each with one of the test access policies. The test user then issues location proximity queries, each with a random user location. The times reported in the results represent the average time to resolve a single location proximity query. All tests were run on a single computer with 16GB memory and an Intel Core i7 2.4GHz CPU running Ubuntu Linux 16.04. We chose to run all tests on a single computer to focus on computation load. In future work we will expand the evaluation to measure communication overhead. Tests were run for varying numbers of AOIs, ranging from 10 to 250, using grid sizes from 100x100, up to 500x500, and various AOI shapes. Figure 5.12 shows the 11 different AOI shapes used

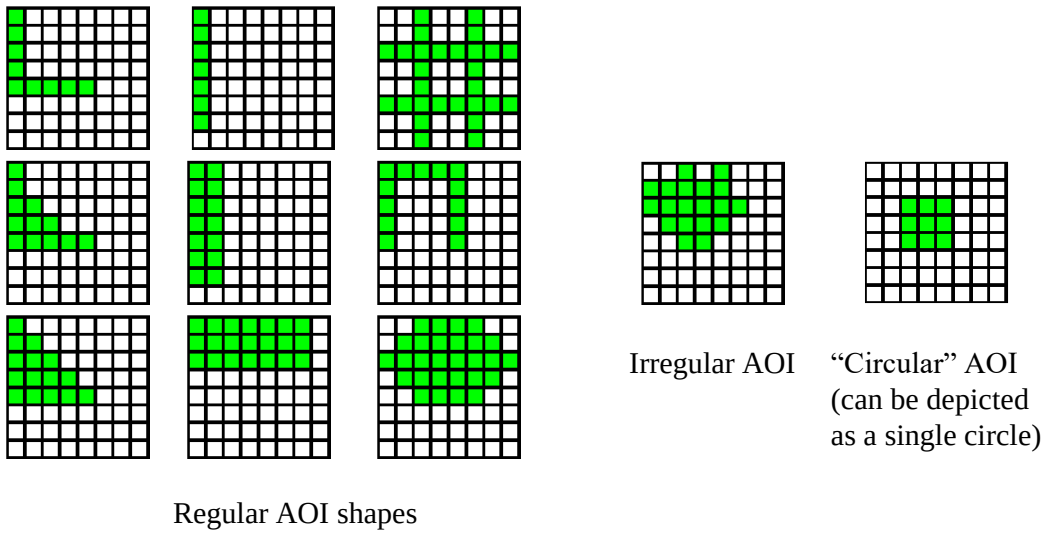
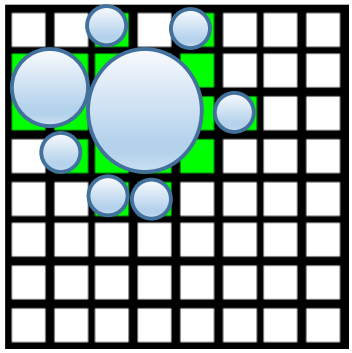


Figure 5.12: AOI shapes used in tests

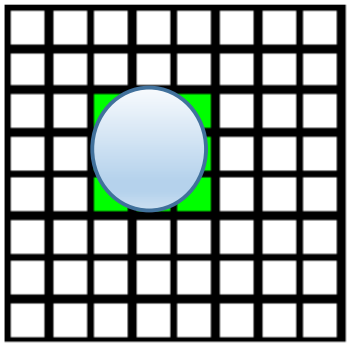
in the performance evaluation. We defined 9 basic shapes, 1 irregular shape, and a standard symmetric AOI represented as a square when using grid cells and as a circle for circle-based AOIs. Figure 5.13 shows how we constructed AOIs using circles to approximate the shapes built from grid cells, and how irregular AOI shapes require more computation due to requirement for more circles to represent each AOI.

Both frameworks were implemented in Python 2.7, using the Charm [2] framework for rapid cryptosystem prototyping. The primary purpose of our prototype implementations is to demonstrate the viability and advantages of our proposed framework and to compare its relative performance to another proposed framework that is similar but less scalable. We found that varying the grid size did not affect the performance in a material way. For this reason, we chose to present results of tests all run using a grid size of 250x250. The unit size can represent any physical distance. In this way, the association of a physical measurement with the unit size defines the precision of the approach. That is, a smaller unit size results in higher precision.



Irregular AOI

Requires 8 circles



Circular AOI
(coverage is similar
to cell-based AOI)

Requires 1 circle

Figure 5.13: Representing AOIs using circles

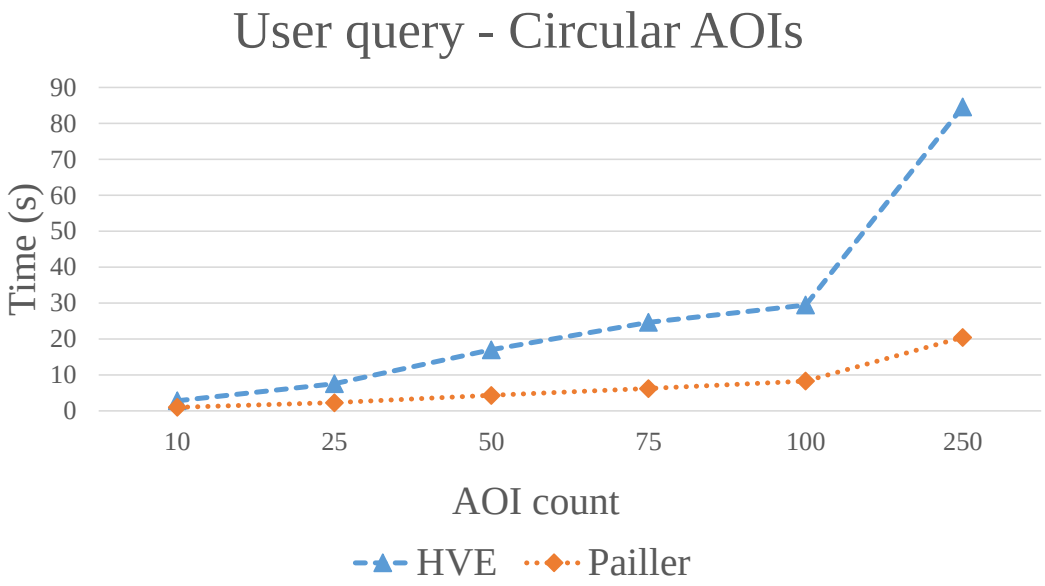


Figure 5.14: Circular AOIs

Our tests show that the choice of HVE for AOI encoding and encryption provides superior scalability over Paillier cryptosystem distance calculations. Li's CP-ABE/Paillier approach works efficiently for circular AOIs, but lacks scalability for more complex AOI shapes. Figure 5.14 shows that CP-ABE/Paillier performs better than our CP-ABE/HVE framework when all AOIs are represented as circles only. This is because our HVE approach requires a collection of cells to define each AOI. Although using Gray codes with token compression to combine multiple tokens, the Paillier distance calculation for a single circle is still more efficient than an HVE token match.

However, when AOI shapes are not circular, the CP-ABE/Paillier approach which requires multiple circles to represent a single AOI exhibits slower performance due to the additional overhead. Our approach can efficiently represent irregular AOI shapes and still provide good performance. Figure 5.15 shows the performance when using AOIs with irregular shapes. The last set of test we ran randomly selected AOI shapes from 10 pre-defined shapes,

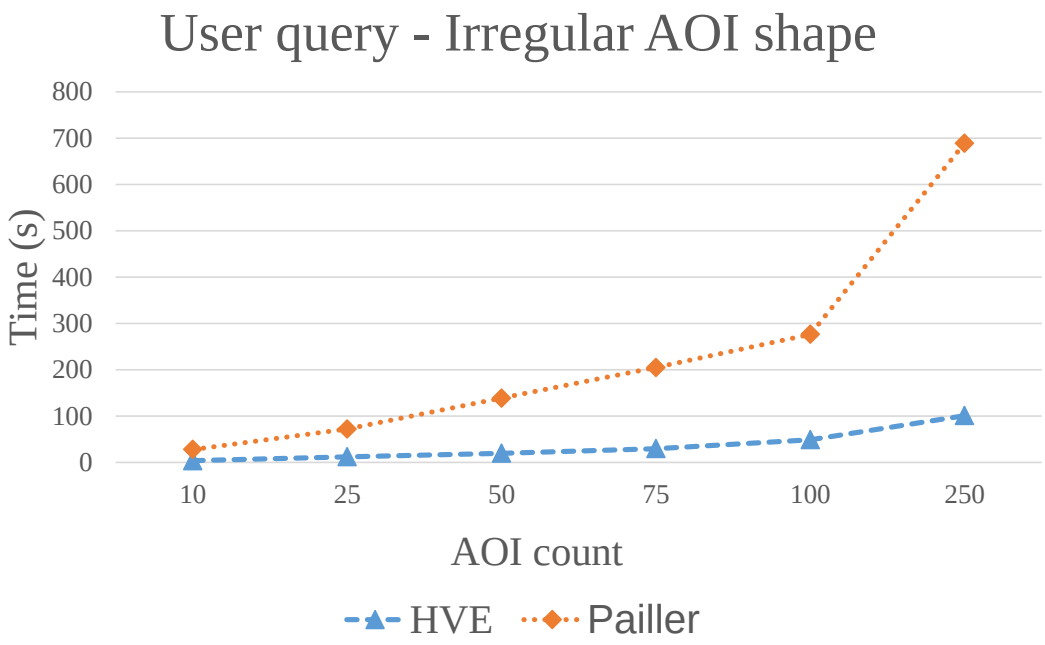


Figure 5.15: Irregular Shaped AOIs

including circular AOIs. Even when selecting some regular shapes, our approach using CP-ABE and HVE performs better than the P-ABE/Paillier approach. Figure 5.16 shows the performance for randomly shaped AOIs.

5.7 Conclusion

We propose a novel framework and protocol based on CP-ABE and HVE that provides MPPD with embedded access control for different classifications of users. With our framework, data owners can define and maintain a single set of AOIs and grant access to AOI information based on user attributes. We implemented our framework and protocol, along with another approach that uses CP-ABE and Paillier cryptosystem, and showed that our approach is more scalable when using AOIs defined as non-circular shapes. Our framework provides a basis for implementers to develop scalable MPPD

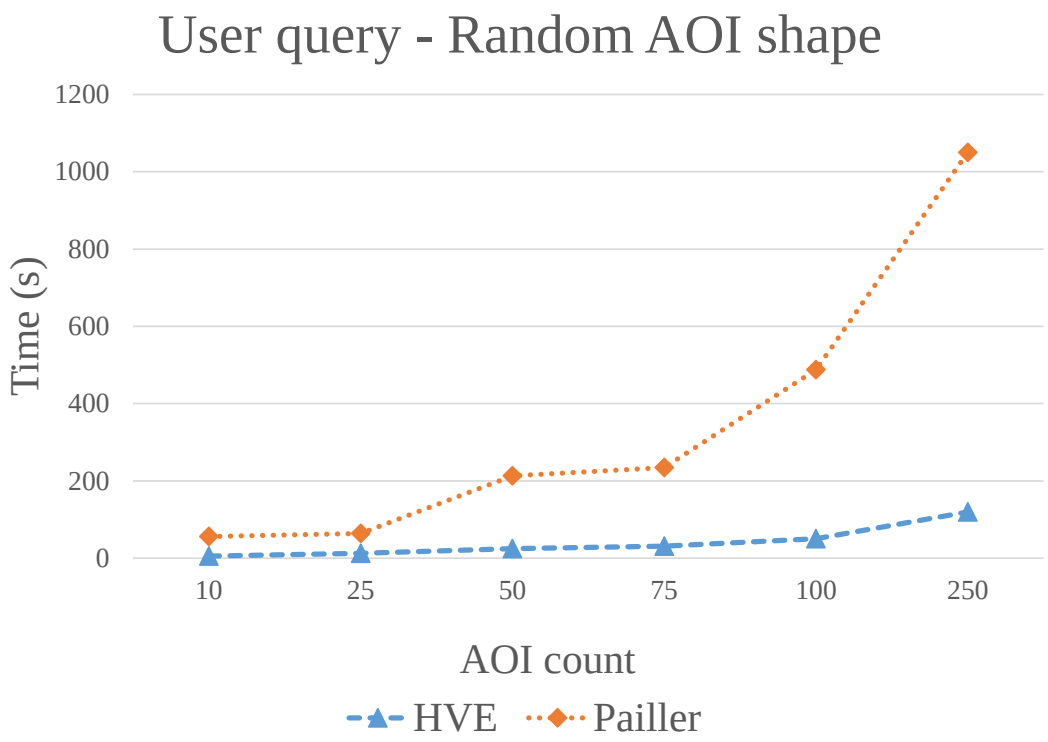


Figure 5.16: Random Shaped AOIs

services that minimizes workload on data owners or users. This approach can provide flexible MPPD services to meet a wide variety of client needs, without requiring a trusted third party to examine user locations to determine proximity.

Chapter 6

Conclusions and Future Work

In this dissertation, we addressed challenges encountered in enforcing both confidentiality of outsourced data to categorical users, and mutually private location privacy detection. Organizations face each of these challenges when designing, developing, and deploying applications that provide location-based or location-sensitive services for enviroments in which privacy for all actors is a design goal and users are authorized to consume data based on descriptive user attributes. Our solutions are viable for inclusion in the design of real world applications and the specific mutually private proximity detection techniques have been shown to exhibit performance characteristics sufficient to support current end user response expectations. We expect that new or upgraded location-sensitive applications can use these solutions to provide privacy to both data owners and users, and provide a flexible methods to control access to various types of data based on data owner policies.

6.1 Summary

6.1.1 Confidentiality in Categorical Settings Contributions

In Chapter 3, we presented the problem of limiting access for data based on data owner requirements to different classifications of users. We showed that

CryptDB provides a partial solution, specifically by providing the ability to search across encrypted data, but does not have the ability to limit access by categorical users. We extended CryptDB by integrating CP-ABE into the storage design. Using CP-ABE, our new framework, ZeroVis, extends all of the benefits of CryptDB to enforce categorical user access, providing confidentiality by user categories, for outsourced data. With ZeroVis, all data inserted into the cloud database carries a data owner provided access policy. This access policy defines the unique collection of user attributes required to decrypt, and thereby consume, the data. A user who requests data but cannot satisfy the access policy for that data is unable to decrypt it and access the resulting plaintext. In a database setting where queries commonly return multiple database rows, ZeroVis will only return rows that both match the query predicate and can be successfully decrypted by the requesting user. The ZeroViz proxy ensures that users are unaware of the existence of, or the number of, rows that match the query predicate but could not be decrypted. As we summarized in Chapter 3, the performance of the initial ZeroVis framework implementation was not optimal and would be inefficient if deployed in its current state. We will leave ZeroVis optimization for future work.

The work on ZeroVis was fundamental for our later work on location privacy in categorical settings. We used the results of the ZeroVis project to strongly influence our later work. We used the concept of access control in categorical settings in our PrivProxABE framework, which we describe in Chapter 6. PrivProxABE combines our work on mutually private proximity detection in general settings with CP-ABE to provide access to location-based information defined by data owner access policies. This enhancement to the frameworks we describe in Chapter 4 allows application designers to define and maintain a single set of AOIs and individual AOI access policies to define user attributes necessary to consume each AOI. Integrating CP-ABE

makes it possible for users to request proximity detection alerts, and only be provided the alerts defined by the AOI data owner, all without having to artificially partition AOIs into sets. The access control is part of the CP-ABE cryptographic technique.

6.1.2 Location Privacy Contributions

In Chapter 4 we introduced the problem of protecting privacy of both data owners and users in location-based services. We described a solution as mutually private proximity detection. Several such solutions have been proposed, and we selected three representative techniques that use encryption to provide proximity detection. As part of our survey we identified a promising approach that addressed "nearness" of vectors of attributes. We found that the approach, Secure k-Nearest Neighbors (SkNN), was novel and warranted further examination. We made modifications to SkNN, extending the original approach, to determine proximity based on distinct locations, as opposed to distance between vectors of attributes. This extension allowed us to include the new SkNN in our implementation assessments.

We implemented three techniques and assessed their performance as we changed input data set sizes. Specifically, we varied the number of AOIs, the size of AOIs, and the size of the overall grid. We summarized our findings and used those to formalize specific recommendations of which technique would likely perform best in a specific deployment setting.

In Chapter 5 we combined our research into categorical access control and mutually private proximity detection to propose a new framework, PrivProx-ABE. This new framework protects the privacy of data owners and users in a location-based services environment, and restricts access to any proximity alert to users based on data owner provided access policies. We found that there is only one other framework similar to ours, but the other framework

exhibits suboptimal performance as it scales. Our framework is designed to provide consistent performance as the input data sizes and users scale, and we presented performance experiment results in Chapter 5.

6.2 Future Work

6.3 ZeroVis Framework

This dissertation describes the initial ZeroVis framework implementation. Future framework changes are necessary to create a more production viable framework. A specific requirement for a trusted AA needs to be included. Although we only generally described the need for the attribute authority, the AA will be an integral component of a completed framework. It will be responsible for authorizing users, securely storing their attributes, and providing the ZeroVis proxy with sufficient authentication information and attributes to properly handle encryption and decryption operations for authorized users. The AA will act as the layer of protection that stops attackers from arbitrarily providing unauthorized attributes to the CP-ABE encryption/decryption functions. The AA will also manage the master key required for encryption/decryption operations.

Additional work is necessary to reduce the storage and computational overhead of CP-ABE. Others have already studied this problem, including Constant-size CP-ABE (CP-ABE) [78] and techniques discussed in [44] and [4]. Future work will also include explore normalizing the CP-ABE ciphertext, which is currently a concatenation of the access policy and the encrypted payload. The access policy comprises over 90% of the ciphertext size. Denormalizing the CP-ABE ciphertext will reduce the storage (and network transmission) requirements for multiple columns that share the same access policy.

6.4 PrivProxABE Framework

As with the ZeroVis framework implementation, our PrivProxABE implementation is an initial solution to mutually private proximity detection in categorical settings. Although our framework performs better than the only other proposed framework that addresses the same problem, we still plan to explore performance enhancements. The key manager needs to be more clearly defined, with a formal API definition that standardizes access to its functionality.

Also, when using the current PrivProxABE design, it is possible for users to simply visit each cell in a grid to determine the locations of all AOIs. Alternatively, users could provide false locations to the service provider to also construct an AOI map. Future extensions of PrivProxABE will need to address the problem of malicious or even curious users that attempt to build an AOI map. We believe that solutions to these problems are feasible and will greatly enhance the utility value of PrivProxABE.

Bibliography

[1] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574. ACM, 2004.

[2] Joseph A Akinyele, Christina Garman, Ian Miers, Matthew W Pagano, Michael Rushanan, Matthew Green, and Aviel D Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[3] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 901–914. ACM, 2013.

[4] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE, 2007.

[5] Alexandra Boldyreva and Virendra Kumar. Provable-security analysis of authenticated encryption in kerberos. *IET information security*, 5(4):207–219, 2011.

[6] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 506–522. Springer, 2004.

[7] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin Heidelberg, 2001.

[8] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography Conference*, pages 535–554. Springer, 2007.

[9] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. A survey of provably secure searchable encryption. *ACM Comput. Surv.*, 47(2):18:1–18:51, August 2014.

[10] Sven Bugiel, Stephen Heuser, and Ahmad-Reza Sadeghi. Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 131–146, 2013.

[11] Luca Calderoni, Paolo Palmieri, and Dario Maio. Location privacy without mutual trust: The spatial bloom filter. *Computer Communications*, 68:4–16, 2015.

[12] M. Carroll, A. van der Merwe, and P. Kotze. Secure cloud computing: Benefits, risks and controls. In *Information Security South Africa (ISSA), 2011*, pages 1–9, Aug 2011.

[13] Melissa Chase and Sherman S.M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 16th*

ACM Conference on Computer and Communications Security, CCS '09, pages 121–130, New York, NY, USA, 2009. ACM.

[14] Xihui Chen, Andrzej Mizera, and Jun Pang. Activity tracking: A new attack on location privacy. In *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 22–30. IEEE, 2015.

[15] Sunoh Choi, Gabriel Ghinita, and Elisa Bertino. Secure mutual proximity zone enclosure evaluation. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, pages 133–142, New York, NY, USA, 2014. ACM.

[16] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

[17] Richard Chow, Philippe Golle, Markus Jakobsson, Elaine Shi, Jessica Staddon, Ryusuke Masuoka, and Jesus Molina. Controlling data in the cloud: Outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, pages 85–90, New York, NY, USA, 2009. ACM.

[18] Transaction Processing Performance Council. Tpc benchmark c, standard specification version 5, 2001.

[19] Dr Deshmukh, Anwar Pasha, Dr Qureshi, et al. Transparent data encryption–solution for security of database contents. *arXiv preprint arXiv:1303.0418*, 2013.

[20] Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Over-encryption: Management of access control evolution on outsourced data. In *Proceedings of the 33rd*

International Conference on Very Large Data Bases, VLDB '07, pages 123–134. VLDB Endowment, 2007.

[21] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[22] R.A. Elmasri and S.B. Navathe. *Fundamentals of Database Systems [With Access Code]*. ADDISON WESLEY Publishing Company Incorporated, 2011.

[23] Yousef Elmehdwi, Bharath K Samanthula, and Wei Jiang. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *2014 IEEE 30th International Conference on Data Engineering*, pages 664–675. IEEE, 2014.

[24] M.R. Farcasescu. Trust model engines in cloud computing. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on*, pages 465–470, Sept 2012.

[25] Kassem Fawaz, Huan Feng, and Kang G Shin. Anatomization and protection of mobile apps location privacy threats. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 753–768, 2015.

[26] Kassem Fawaz and Kang G Shin. Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 239–250. ACM, 2014.

[27] Luca Ferretti, Michele Colajanni, Mirco Marchetti, and Adriano Enrico Scaruffi. Transparent access on encrypted data distributed over multiple cloud infrastructures. In *CLOUD COMPUTING 2013, The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 201–207, 2013.

[28] Amos Fiat and Moni Naor. Broadcast encryption. In DouglasR. Stinson, editor, *Advances in Cryptology CRYPTO 93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer Berlin Heidelberg, 1994.

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

[30] Gabriel Ghinita and Razvan Rughinis. A privacy-preserving location-based alert system. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 432–435, New York, NY, USA, 2013. ACM.

[31] B. Gowrigolla, S. Sivaji, and M.R. Masillamani. Design and auditing of cloud computing security. In *Information and Automation for Sustainability (ICIAFs), 2010 5th International Conference on*, pages 292–297, Dec 2010.

[32] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM.

[33] Per Hallgren, Martın Ochoa, and Andrei Sabelfeld. Maxpace: Speed-constrained location queries.

[34] Fei Han, Jing Qin, and Jiankun Hu. Secure searches in the cloud: A survey. *Future Generation Computer Systems*, 62:66 – 75, 2016.

[35] Urs Hengartner. Hiding location information from location-based services. In *2007 International Conference on Mobile Data Management*, pages 268–272. IEEE, 2007.

[36] Luan Ibraimi, Milan Petkovic, Svetla Nikova, Pieter Hartel, and Willem Jonker. Ciphertext-policy attribute-based threshold decryption with flexible delegation and revocation of user attributes. *Univeristy of Twente, Tech. Rep*, 2009.

[37] Wayne Jansen, Timothy Grance, et al. Guidelines on security and privacy in public cloud computing. *NIST special publication*, 800:144, 2011.

[38] K.M. Khan and Q. Malluhi. Establishing trust in cloud computing. *IT Professional*, 12(5):20–27, Sept 2010.

[39] Ali Khoshgozaran and Cyrus Shahabi. Private buddy search: Enabling private spatial queries in social networks. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 166–173. IEEE, 2009.

[40] Ali Khoshgozaran and Cyrus Shahabi. Private information retrieval techniques for enabling location privacy in location-based services. In *Privacy in Location-Based Applications*, pages 59–83. Springer, 2009.

[41] Hyeong-Il Kim and Jae-Woo Chang. k-nearest neighbor query processing algorithms for a query region in road networks. *Journal of Computer Science and Technology*, 28(4):585–596, 2013.

[42] Hyeong-Il Kim, Seungtae Hong, and Jae-Woo Chang. Hilbert curve-based cryptographic transformation scheme for spatial query processing on outsourced private data. *Data & Knowledge Engineering*, 2015.

[43] Hyeong-Il Kim, Yong-Ki Kim, and Jae-Woo Chang. A grid-based cloaking area creation scheme for continuous lbs queries in distributed systems. *JoC*, 4(1):23–30, 2013.

[44] Jongkil Kim, Willy Susilo, ManHo Au, and Jennifer Seberry. Efficient semi-static secure broadcast encryption scheme. In Zhenfu Cao and Fangguo Zhang, editors, *Pairing-Based Cryptography Pairing 2013*, volume 8365 of *Lecture Notes in Computer Science*, pages 62–76. Springer International Publishing, 2014.

[45] G. Kulkarni, N. Chavan, R. Chandorkar, R. Waghmare, and R. Palwe. Cloud security challenges. In *Telecommunication Systems, Services, and Applications (TSSA), 2012 7th International Conference on*, pages 88–91, Oct 2012.

[46] Mehmet Kuzu, Mohammad Saiful Islam, and Murat Kantarcioglu. Distributed search over encrypted big data. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, CODASPY '15, pages 271–278, New York, NY, USA, 2015. ACM.

[47] Stephen Lawson. Ten ways your smartphone knows where you are, 2016.

[48] Seung-Jin Lee, Eui-Nam Huh, et al. Shear-based spatial transformation to protect proximity attack in outsourced database. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1633–1638. IEEE, 2013.

[49] Wei-Bin Lee and Chien-Ding Lee. A cryptographic key management solution for hipaa privacy/security regulations. *Information Technology in Biomedicine, IEEE Transactions on*, 12(1):34–41, Jan 2008.

[50] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. Scalable and secure sharing of personal health records in cloud computing us-

ing attribute-based encryption. *Parallel and Distributed Systems, IEEE Transactions on*, 24(1):131–143, Jan 2013.

[51] Xiang-Yang Li and Taeho Jung. Search me if you can: privacy-preserving location query service. In *INFOCOM, 2013 Proceedings IEEE*, pages 2760–2768. IEEE, 2013.

[52] Rongxing Lu, Xiaodong Lin, and Xuemin Shen. Spoc: A secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency. *IEEE Transactions on Parallel and Distributed Systems*, 24(3):614–624, 2013.

[53] Diana Stefania Maimut, Alecsandru Patrascu, and Emil Simion. Homomorphic encryption schemes and applications for a secure digital world. *Journal of Mobile, Embedded and Distributed Systems*, 4(4):224–232, 2012.

[54] T. Mather, S. Kumaraswamy, and S. Latif. *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. Theory in practice. O'Reilly Media, 2009.

[55] Ed Novak and Qun Li. Near-pri: Private, proximity based location sharing. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 37–45. IEEE, 2014.

[56] Paolo Palmieri, Luca Calderoni, and Dario Maio. Spatial bloom filters: enabling privacy in location-aware applications. In *International Conference on Information Security and Cryptology*, pages 16–36. Springer, 2014.

[57] Duong-Hieu Phan, David Pointcheval, SiamakF. Shahandashti, and Mario Strefler. Adaptive cca broadcast encryption with constant-size

secret keys and ciphertexts. *International Journal of Information Security*, 12(4):251–265, 2013.

[58] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 85–100, New York, NY, USA, 2011. ACM.

[59] B. T. Prasanna and C. B. Akki. A comparative study of homomorphic and searchable encryption schemes for cloud computing. *CoRR*, abs/1505.03263, 2015.

[60] A. Roxin, J. Gaber, M. Wack, and A. Nait-Sidi-Moh. Survey of wireless geolocation techniques. In *Globecom Workshops, 2007 IEEE*, pages 1–9, Nov 2007.

[61] Zhidong Shen and Qiang Tong. The security of cloud computing system enabled by trusted computing technology. In *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, volume 2, pages V2–11–V2–15, July 2010.

[62] Michael G Solomon, Vaidy Sunderam, and Li Xiong. Towards secure cloud database with fine-grained access control. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 324–338. Springer, 2014.

[63] Michael G Solomon, Vaidy Sunderam, Li Xiong, and Ming Li. Enabling mutually private location proximity services in smart cities: A comparative assessment. In *Smart Cities Conference (ISC2), 2016 IEEE International*, pages 1–8. IEEE, 2016.

[64] Michael G Solomon, Vaidy Sunderam, Li Xiong, and Ming Li. Privacy preserving location proximity detection in categorical settings. under review, 2016.

[65] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[66] Stephen Tu, M. Frans Kaashoek, Samuel Madden, and Nickolai Zeldovich. Processing analytical queries over encrypted data. *Proc. VLDB Endow.*, 6(5):289–300, March 2013.

[67] Jaroslav Šeděnka and Paolo Gasti. Privacy-preserving distance computation and proximity testing on earth, done right. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, pages 99–110, New York, NY, USA, 2014. ACM.

[68] Boyang Wang, Ming Li, and Haitao Wang. Geometric range search on encrypted spatial data. *IEEE Transactions on Information Forensics and Security*, 11(4):704–719, 2016.

[69] Boyang Wang, Ming Li, Haitao Wang, and Hui Li. Circular range search on encrypted spatial data. In *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 182–190. IEEE, 2015.

[70] Yonghui Xiao and Li Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1298–1309. ACM, 2015.

[71] Man Lung Yiu, Ira Assent, Christian S Jensen, and Panos Kalnis. Outsourced similarity search on metric data assets. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):338–352, 2012.

[72] Man Lung Yiu, Gabriel Ghinita, Christian S Jensen, and Panos Kalnis. Enabling search services on outsourced private spatial data. *The VLDB Journal*, 19(3):363–384, 2010.

[73] Min Yoon, Hyeong-il Kim, Miyoung Jang, and Jae-Woo Chang. A new spatial transformation scheme for preventing location data disclosure in cloud computing. *International Journal of Data Warehousing and Mining (IJDWM)*, 10(4):26–49, 2014.

[74] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, March 2010.

[75] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Attribute based data sharing with attribute revocation. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 261–270, New York, NY, USA, 2010. ACM.

[76] Qingji Zheng, Shouhuai Xu, and Giuseppe Ateniese. Vabks: Verifiable attribute-based keyword search over outsourced encrypted data. Cryptology ePrint Archive, Report 2013/462, 2013. `http://eprint.iacr.org/`.

[77] Lan Zhou, Vijay Varadharajan, and Michael Hitchens. Enforcing role-based access control for secure data storage in the cloud. *The Computer Journal*, 54(10):1675–1687, 2011.

[78] Zhibin Zhou and Dijiang Huang. On efficient ciphertext-policy attribute based encryption and broadcast encryption: Extended abstract. In *Pro-*

ceedings of the 17th ACM Conference on Computer and Communications Security, CCS '10, pages 753–755, New York, NY, USA, 2010. ACM.

[79] Xukai Zou, Yuan-Shun Dai, and E. Bertino. A practical and flexible key management mechanism for trusted collaborative computing. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 538–546, April 2008.