Imperial College
London

PrivMarket: Privacy and Anonymity in Location-Based User Data Markets

Author:
Corina Niculae

Supervisor:
Dr. Naranker Dulay

Second Marker:
Dr. Emil Lupu

June 2016

Submitted in part fulfillment of the requirements for the degree of
Master of Engineering in Computing of Imperial College London

Abstract

Online service providers take location data for granted, while enforcing users to agree to inflexible and intricate policies. This leads to location privacy issues that can imperil the Internet users' comfort, welfare, and even safety. One possible solution to this is represented by location data markets which empower users to regain ownership of their data.

We analyse the current state of art that relies on cryptographic solutions when tackling the problem of location privacy and anonymity. Fully homomorphic encryption, a cryptographic paradigm that allows the computation of any function over encrypted data, is still prohibitively expensive; however, recent advancements led to specialised encryption schemes that can apply particular classes of functions. We propose a new model, **PrivMarket**, building upon the premise of running statistical computations over spatial traces.

We formalise the model by using two novel frameworks targeted to the development of location-aware services. We then implement a location data market for the Transport of London (TFL) travelers, as part of a case study. The proof of concept supports a basic set of queries that have been run on an artificial data set we have built, and integrates with a new piece of technology, CryptDB. We provide an in-depth overview of **PrivMarket** and CryptDB, including their limitations and potential future developments.

Acknowledgments

Firstly, I would like to thank my project supervisor, Dr. Naranker Dulay, for his sincere enthusiasm, guidance and immense support throughout the past months that definitely helped me steer the project into the right direction, and inspired me to tackle challenging problems.

I would like to thank all my wonderful friends, for their fruitful conversations, as well as their constant belief in my success. Finally, I would like to thank my family and my beloved Dan Florescu, for their everlasting and unconditional support.

Contents

Chapter 1

Introduction

1.1 Motivation

Current technological advancements make it feasible to spatially identify and track mobile devices. Examples include specialised localisation systems, such as the Global Positioning System (GPS), or practices that infer location from existing infrastructures, like the Global System for Mobile communication (GSM) network. It has become a commodity for online services to impose such positioning requirements, especially when targeting mobile end-users. Consequently, new areas of study and issues have been exposed, the most critical one being location privacy.

Location traces are some of the most valuable digital crumbs one leaves behind when using an online product or service, because they leak out information on the user. Under the false premise of free-to-use Internet-based services, their underlying infrastructures quietly harvest your location traces. Any place visited while carrying a smart-phone (even without an Internet connection) gives away information on the owner, whether it is a personal preference (e.g., going to an Italian restaurant), or a medical condition (e.g., going to an emergency clinic). In the wrong hands, this information may be used against the user, leading to inconvenience, blackmail, and even harassment.

However, the online content providers back-up their practices by contorted policies and hard-to-follow regulations that confuse the potential user. On top of that, their policies' inflexibility offers little to no usage alternatives of the product. This produces friction with the netizens[1] that want to enjoy the benefits of location-aware services, but also secure their mobility information.

Given the complexity of the issue of location privacy, one possible solution relies on building technological tools and systems that might provide some privacy guarantees. There has been a recent rise of **user data markets**, platforms on which users decide who to share their information with, including their location, and, in exchange, receive remuneration for their participation. In this context, we refer to **producers** as users that make their location data available to be used within the market; we denominate **consumers** as parties interested in computing some statistical functions over the collected data. The current underlying market model does not provide enough privacy guarantees, putting the system at risk for possible adversary attacks.

We embark on a technological quest to help finding a suitable solution for guaranteeing the producers' location privacy and anonymity within data markets[2], while preserving the information utility for the consumers.

[1] An umbrella term for the entire spectrum of Internet users.

[2] Unless specified, the following terms will be used interchangeably throughout the paper: 'user location data market', 'location data market', and 'data market'.

1.2 Objectives

This project would like to investigate the privacy and anonymity aspects of data markets, and contribute toward building such a platform. There are existing system proposals that offer privacy guarantees, but they are highly restricted to particular scenarios. We delve into the problem of storing location data while preserving its owner's privacy, as well as the data's utility. The main objective is to better understand the opportunities that arise in this field, and the limitations imposed by the problem. We aim to attach to our studies a pragmatic proof of concept in order to observe their feasibility.

Another objective is to consider designing a user reward system for data markets. Since financial rewards seem the most sensible, as well as the most attractive, a challenging objective is to develop a payment system that can reward the users without exposing their real identity. The market, on behalf of the consumer, needs to know how to reach the producer in order to remunerate their contribution. The possibility of designing and implementing such a system is entailed by the intrinsic considerations and limitations. For instance, the reward criteria can be based on the quality of the provided data, or only on the overall participation of the user. The first needs a more restrictive system that assigns a quality score to each user, which might lead to a weaker privacy model; meanwhile, the second can be handled separately from the data market model.

1.3 Contributions

The main contributions of the project are based on the following considerations:

1.3.1 *PrivMarket*: Location Data Market Model

A real-world data market must be backed by a well-defined ideal model that provides a holistic approach to the problem of user anonymity vs. data utility. For this, we studied the mindset of the related state of art (**chapter 3**, together with **chapter 2 that provides the needed background blocks**). After a discussion on each system's novelty and limitations, we loosen their initial considerations, and propose $PrivMarket$. This model's goal is to run statistical computations over location traces. The data market system allows smart-phone users (producers) to make their locations available to entities interested in computing a statistical queries over their input (consumers, or aggregators).

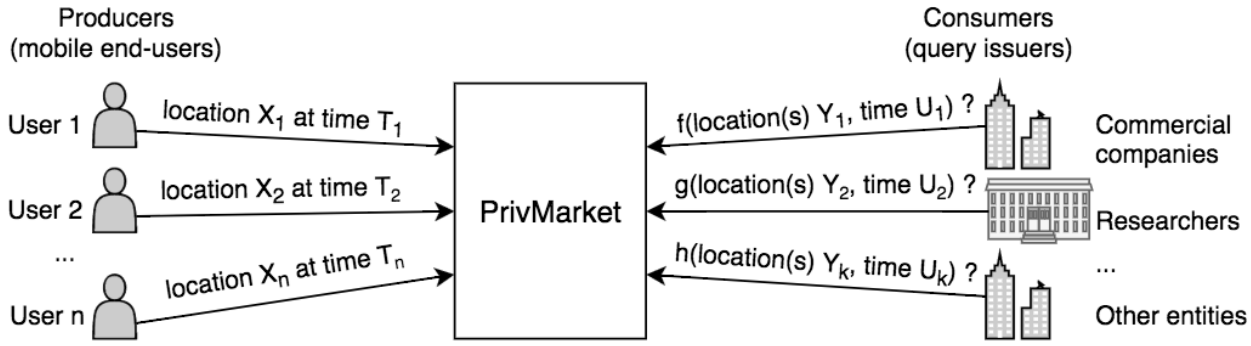

Figure 1.1: Overview of stakeholders in $PrivMarket$. The data market has a set of mobile end-users that produce the data and make it available to some extent to $PrivMarket$. We later on expand on the underlying model. There is also a set of entities interested in running statistical functions over the users' locations. They include, but are not restricted to, commercial companies, researchers, local authorities, and so on.

The involved stakeholders are pictured in figure 1.1. Our input unit is given by the outbound (or inbound, respectively) journey traveled throughout a day, up to (or from, respectively) furthest point

from the user's home location. We analyse the model in the context of an adversary which aims to infer information on the users of the data market by using the system. The project extends the existing scale for quantifying the user anonymity that is introduced in [PBBL11]. **Chapter 4** provides this, as well as the design modeling of $PrivMarket$.

1.3.2 Case Study: Transport of London (TFL) Data Market

We provide an application of our model, in the context of leveraging smart cities' opportunities. A smart city represents an urban development that securely integrates multiple information and technology (ICT) solutions in order to better manage the city's assets, such as local departments information systems, transportation systems, utility supply networks, waste management, law enforcement, and other community services. We focus on the TFL infrastructure, particularly the tube, overground and DLR services, and implement a proof of concept for a location data market aimed to the everyday users of the aforementioned traveling networks (**chapter 5**).

Within the prototype, we focus on basic statistical queries that can be used by TFL to better understand the usage of their services. We gauge its limitations and provide a discussion on its real-world integration in **chapter 6**.

Lastly, the report gives an extensive cover of possible extensions and future work opportunities in **chapter 7**, and ends with the conclusions in **chapter 8**.

Chapter 2

Background

We provide an introduction into location awareness and location privacy as our departure point. This is followed by an overview of some cryptographic blocks to which we will refer in the subsequent chapters.

2.1 Location Awareness

The **location data** of an individual is a (sub)set of their spatial positions over some time period. The granularity of the collected information might vary, depending on the case. **Location awareness** encompasses any mobile device that can **actively** or **passively** determine its location [Leo98].

As [Leo98] notices, location awareness is often directed **inwards** - a mobile user would like to know their location at all times, and maybe even have historical records of it. This has become more relevant with the increase of accessibility to mobile devices (e.g. smart-phones). Location awareness is also directed **outwards**, when (close) parties would like to know another's location. For instance, parents might want to know where their children are, spouses might want to share their location with each other, or estate agents may need to know what property their team members are visiting.

Location aware systems can determine the whereabouts of their users, and even track them over time. They include, but are not limited to location-based systems, as location data may have different roles within the service:

- It can be the goal of the system, as in the case of **geo-location services**, **mobile sensing** or **location sharing applications**.
- Location information can be used for service partitioning; this is the case of **location-based services**, where the content is customised in accordance with the user's location.
- User location can also be treated as second-class citizen data; while it does not serve the main objective of the service, it increases the user experience. For example, this is the case of online social networks (Facebook[1], Google+[2], etc.) that can also operate without the user's location.

Since there exists a demand and a set of offered services on the market of location-aware services, the issue of location privacy naturally arises. The reuse of current networks and infrastructures for location aware applications comes at a cost.

[1] www.facebook.com
[2] www.plus.google.com

2.2 Location Privacy

Location data is subject to semantic inference, as personal preferences or activities are correlated with one's location. For instance, Alice[3] going to a specific restaurant leaks her gastronomic preferences. But, some inferred information might be too sensitive, such as Alice's location in an abortion treatment clinic. Mining location data can unveil private facts about an individual, such as their religious or political affiliations, their medical conditions, or their business and social connections.

In the wrong hands, this information can lead to reputation or relationship damage, personal harassment (for instance, religious members of Alice's community may not be happy with her medical choices), or any kind of attack toward the individual in question, or their family and friends. **People need location privacy** - they need to be in control of their location data and be able to specify when to share it and who to share it with. According to [DK06], location privacy is the right of individuals to determine for themselves when, how, and to what extent location information about them is communicated to others.

Location aware services take user location data for granted, offering little transparency or operational alternatives to their services. Their business model mostly relies on offering free usage of their service against the right of accessing and collecting all location data generated on their platform [DV11, Pol15]. These practices are hidden behind intricate policies and regulations on storing and sharing the aggregated data. Consequently, online users unwillingly leave their spatial traces to companies which generate revenue by selling or using them for targeted advertising. Users have no control on what data mining practices run over their location information.

Furthermore, storing location data comes with the risk of any sensitive data storage. An adversary can take advantage of a possible security breach. There are numerous cases of security breaches, as listed in [lis, Har15]. A poor implementation of the module responsible for collecting location can lead to unauthorised releases to third parties, like suggested in [ZDG+15]. A location data leakage on which the adversary applies data mining or other information inference techniques can harm the individuals, as previously presented.

If combined with other seemingly anonymous data, [Swe00] showed that only three categories of user data, **(5-digit zip code of home location, gender, date of birth) uniquely identify 87% (216 million our of 248 million)** of the population that was residing in the United States in 1990. More recent studies focus on information inference solely from location data. In [Kru07], 4 different algorithms for inferring one's home location are introduced. In the experiment, **the home location of all 172 subjects was recovered with a median error below 60 meters, provided 2 weeks record of their GPS location**. [GP09] expanded on this experiment, by taking the conjoined pair (home location, work location) and proved that the knowledge of one's home and work location, with a granularity of a consensus block can uniquely identify the majority of the United States working population. In [dMHVB13], **hourly locations collected via the carrier's antennas for 15 months made it possible to uniquely identify 95% of the user traces, given 1.5M users**.

Prohibiting location collection is impossible at this point, given the underlying economic models and the multitude of application features based on user location [Pie]. Given the current advancements of data mining and other information inference methodologies, the personally identifiable information (PII) set needs to be expanded with location traces. However, location information still needs to be available for online applications and services to carry out their operational activities for the users' benefits.

[3]Throughout the report, we are going to use Computing literature's generic names for illustrating examples, such as *Alice*, *Bob*, etc.

2.3 Data Markets

User data markets are becoming a trending topic among smart-phone users. There are various examples of user data markets, such as DataCoup [dat16], CitizenMe [cit15], people.io [peo16], or digi.me [dig16].

All these user data markets are centralised, collecting the user data (including location) to an internal centralised database. Third party entities perform computations over the collected data for demographic analysis that may later on influence targeted advertising or research studies. Out of the pool of existing user data markets, a concerning number of them are vaguely treating the issue privacy or anonymity of the collected data with the user (i.e. data producer), while the rest of them provide assurances that the collected data is encrypted and stored behind multiple firewalls, or that is securely stored and not shared with third parties. This still makes them attractive for an adversary attack. Even if they do provide sufficient security guarantees for traditional personally identifiable information (PII), **they need a feasible solution for keeping the collected location traces both anonymous and useful** such that third parties can use them in a controlled environment.

Since (partially) homomorphic encryption schemes (see subsection 2.4.2) are still under development, running computations on encrypted data is not a standard yet. In order to run statistical computations (for example, the average number of people between 20 and 30 that live in South Kensington), one needs to decrypt the data while the computation is pending. Also, most of these systems anonymize the data after it was received, so the user needs to have a high level of trust in these data markets.

The picture below shows DataCoup's home view, where multiple classes of user data are requested, such as financial, social, or health information is requested. Location data is also harvested by connecting the user's Foursquare[4] a location-based search and discovery application that collects data on various locations from the user within the platform.

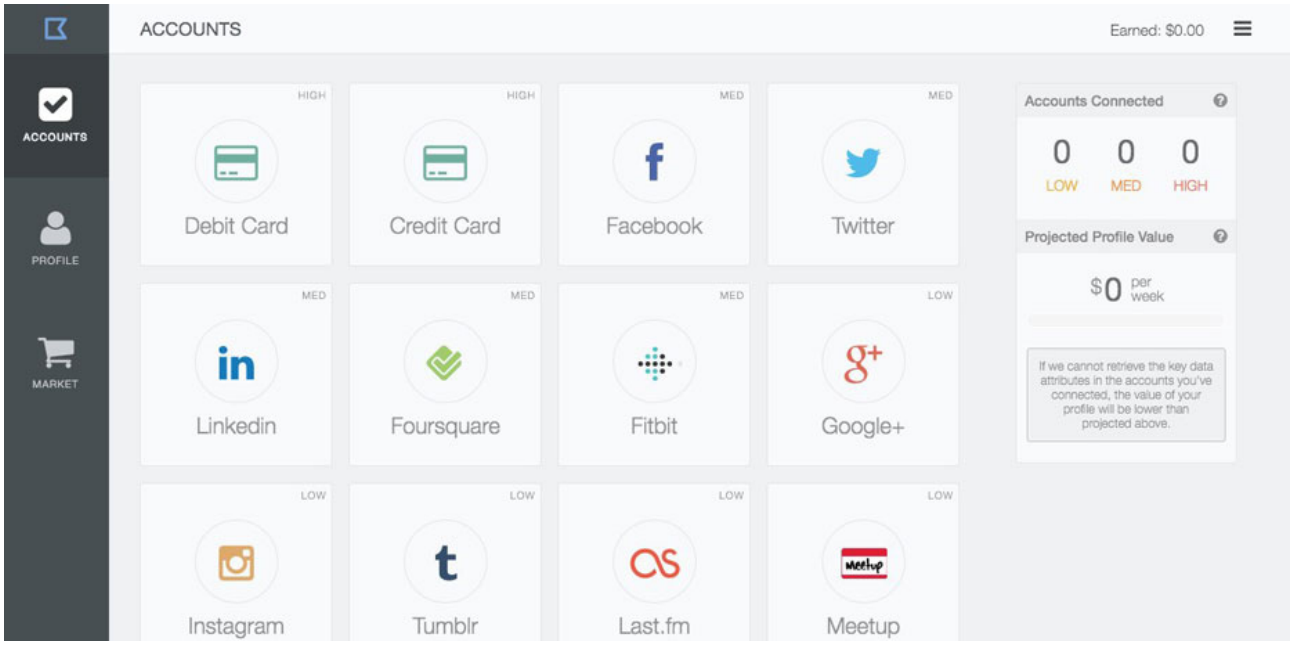

Figure 2.1: Home view of a logged user that provides their data to DataCoup. Image courtesy of [dat16].

Data markets can be leveraged as standard practices for managing user data, provided that some privacy and anonymity guarantees are offered. Existing business models can integrate a data market

[4] www.foursquare.com/

to replace the current data collection practices, while adapting their service offering - **free usage against participation into the data market**.

The current situation clearly emphasises the need of a decentralised user data market that preserves its users anonymity and privacy.

2.4 Cryptographic Blocks

In this section we present some cryptographic concepts that represent the base of the systems related to PrivMarket, in order to better understand their goals and characteristics. The reader may skip this section and refer back to it as indicated in the following chapters, or read it now in its entirety. The colour mapping in this chapter follows [Sma03] considerations, where blue signifies public information, and red represents secret information.

2.4.1 Commitment Schemes

A commitment scheme is a two-phase protocol between two parties, Alice (the sender) and Bob (the receiver):

1. Initially, Alice holds a message m (where $m \in \mathcal{M}$, the set of all possible messages in the corresponding application space), and would like to show Bob her binding of m (i.e., that she sticks or **commits** to the value) over a time period, without revealing the actual value. Thus, she produces a random value, r, that is initially secret, and sends Bob a hash of r and m, representing her cryptographic commitment.

$$\mathsf{Alice} \longrightarrow \mathsf{Bob} : H(r||m)$$

2. Later on, when Alice wants or has to reveal (or **decommit**) the initial message to Bob, she sends him the values that composed the result of the previous hashing. Bob can now check that was the initial value, as the hash function was public.

$$\mathsf{Alice} \longrightarrow \mathsf{Bob} : r, m$$

To be noted that the hash function has to be second image collision resistant, such that no party can cheat in the protocol. This leads to defining **binding** and **concealing** as the principal characteristics of quantifying and classifying a commitment scheme. According to [Sma03], binding can be information theoretically (or computationally, respectively), if Alice (the sender) would never be able to find a second image of the hash function in order to cheat in the protocol, given unlimited computational power (or limited one, respectively). Ibidem, concealing would be information theoretically (or computationally), if Bob would never be able to find the initial values of Alice's commitment, given unlimited computational power (or limited one, respectively).

2.4.2 Homomorphic Encryption

Generally, encryption schemes offer anonymity against little usability of the encrypted data. Homomorphic encryption addresses this issue.

Homomorphic encryption (HE) is an encryption scheme where the operation over the encrypted messages can be re-written as the encryption of the result of a binary operator applied over their two correspondent plain messages. Formally, given an encryption scheme $(\mathbb{P}, \mathbb{C}, \mathbb{K}, e_k(\cdot), d_k(\cdot))$, the scheme

is a homomorphic encryption iff[5] the following property is satisfied[Sma03]:

$$E_k(m_1) \odot E_k(m_2) = E_k(m_1 \diamond m_2), \forall\ m_1,\ m_2 \in \mathbb{P}$$

To be noted that in a **homomorphic commitment scheme**, the hash function has the previously defined property.

The $\diamond$ operator gives two main classes of HE, namely **additively homomorphic encryption** (AHE), when $\diamond$ is the additive operator, $+$, and **multiplicatively homomorphic encryption** (MHE), when $\diamond$ is the multiplication operator $*$. There have been proposed a variety of HE systems, out of which we mention the following:

Paillier Cryptosystems

A Paillier Cryptosystem (PC) [Pai99] is an asymmetric key encryption scheme[6]. It is an additive HE. Let there be two parties, Alice and Bob. Then, the PC algorithm is as following:

1. **Key Generation**
   (a) Bob generates two large safe primes, p and q, such that $gcd(pq, (p-1)(q-1) = 1$.
   Now, let $n = pq$ and $\lambda = (p-1)(q-1)$.
   (b) Furthermore, let $g = n + 1$, and $\mu = \lambda^{-1}\ mod\ n$.
   (c) Now Bob has a (public key, private key) pair, where the public key is $PK = (n, g)$ and the private key is $SK = (\lambda, \mu)$
2. **Encryption** For any message to be send, $\forall\ m \in \mathbb{Z}_{\ltimes}$, Alice does the following:
   (a) Selects a random $r \in \mathbb{Z}^*_{\ltimes}$.
   (b) Alice produces the ciphertext, $c = E(m) = g^m * r^n\ mod\ n^2$.
3. **Decryption** For any ciphertext, $\forall c \in \mathbb{Z}^*_{\ltimes}$, Bob recovers the original message by performing the following operation:

$$m = L(c^{\lambda}\ mod\ n^2) * \mu\ mod\ n, \text{ where:}$$
$$L(u) = \frac{u-1}{n}$$

The proof of the AHE property over Paillier cryptosystems is not subject of this report.

Pedersen's HE scheme

Given an encryption scheme $(\mathbb{P}, \mathbb{C}, \mathbb{K}, e_k(\cdot), d_k(\cdot))$, Pedersen's scheme [Ped91] is another AHE with the following properties:

$$e_k(m_1) * e_k(m_2) = e_k(m_1 + m_2)$$
$$d_k(m_1 + m_2) = d(m_1) * d(m_2)$$

For more information on the scheme, please see its introductory paper, [Ped91], as going in more depth is not under the scope of this project.

[5] 'iff' will denote from now on 'if and only if'.

[6] An asymmetric encryption is a public-key encryption that uses a pair of (public key, private key) for encryption, and, respectively, decryption operations.

2.4.3 Secure Multi-party Computation

A secure multi-party computation (SMC) is a protocol between two or more parties that have some privacte data and collaborate in order to compute a function over everyone's data while keeping it private. The most famous example is Yao's millionaires problem[Yao82].

Obviously, at the end of the protocol, each party will know the final result, together with their relative position against the final outcome. If this is the only information leaked at the end of the SMC protocol, it is considered to be **secure**. Since the trivial solution to this problem is to take a trusted third-party entity that collects all inputs and computes the result, an SMC protocol is considered **correct** if it behaves equivalently to this trusted third-party protocol.

There are two underlying threat models of SMCs, namely honest-but-curious parties, that are assumed to behave correctly, but are curious in learning more on the other parties, and malicious parties, respectively, that try to subvert the entire protocol. There are two SMC protocol families, Yao's garbled circuit [KS08], and secret sharing schemes [Bri89, BL90].

2.4.4 Oblivious Transfer

Given two parties, Alice (sender) and Bob (verifier), in the 1-out-2 oblivious transfer (OT), Alice would like to allow Bob to choose between one of her secret messages, m_0 and m_1. He knows that he has a bit-available choice (i.e. Bob will have b, where b can be 0 or 1), but he wouldn't want Alice to know what his choice was. At the same time, Alice does not want to reveal the unpicked choice.

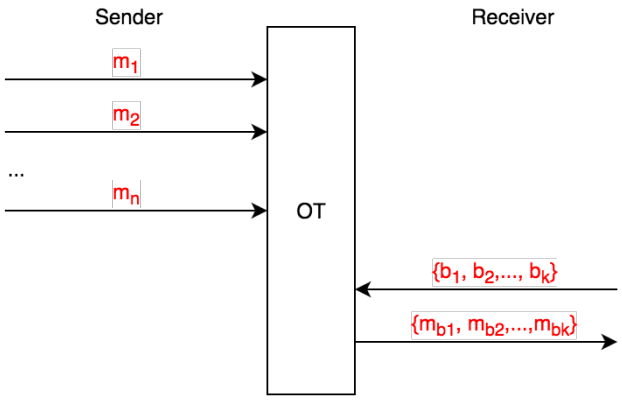

Figure 2.2: A k-out-of-n OT.

An oblivious transfer with a general k-out-of-n goal (as illustrated in 2.2 entail two parties, Alice and Bob (receiver), with the following data and requirements:

- Alice has n messages, $\{m_1, m_2, ..., m_n\}$.
- Bob has a set of choices $\{b_1, b_2, ..., b_k\}$, such that $\forall\ b_i \in \{1, 2, ..., n\}$, representing k message choices from Alice's initial set.
- However, Alice does not want to share all message options with Bob before his choice, only his final chosen messages. Similarly, Bob does not want Alice to find out his k message picks.

For more information on this, please refer to [Rab05].

2.4.5 Zero-Knowledge Proof

Abstractly, a zero-knowledge proof (ZKP) is a method of demonstrating the validity of a statement without revealing the actual statement[Gol98]. Let there be two parties, a prover (Peggy) and a verifier

(Victor), and a challenge to be solved, $\mathcal{C}$.

A ZKP protocol is composed by three stages, namely commitment, challenge and response, respectively. In [Sma03], ZKP is simply illustrated by the problem of graph isomorphism: given two graphs, G_1 and G_2 as illustrated, Peggy needs to prove to Victor that she is able to find a graph isomorphism, without revealing the actual solution. We assume she manages to get $\sigma = (1, 2, 4, 3)$ as a solution to the challenge. Now, she chooses another isomorphism, say $\phi = (1, 2)$ and computes H. We now have the following properties:

$$\begin{aligned} \sigma : G_1 &\longrightarrow G_2 \\ \phi : G_2 &\longrightarrow H \\ \phi \circ \sigma = \sigma(\phi) : G_1 &\longrightarrow H \end{aligned}$$

She now sends H to Victor, representing her *commitment* of solving the challenge. Then Victor challenges her, by choosing G_1 or G_2. For any possible choice G_b, where $b \in 1, 2$, Peggy needs to prove that she found an isomorphism from G_b to H. If b is 2, she simply responds to Victor with ϕ. In case Victor requested challenge 1, Peggy provides the composition of ϕ and σ, $\phi \circ \sigma$. Victor has only one challenge choice available; otherwise, he would be able to learn Peggy's solution. In order for Peggy to earn Victor's trust, they run this protocol for a multiple number of times, each time Peggy providing a new commitment (i.e. a new graph H' in this case) to be challenged.

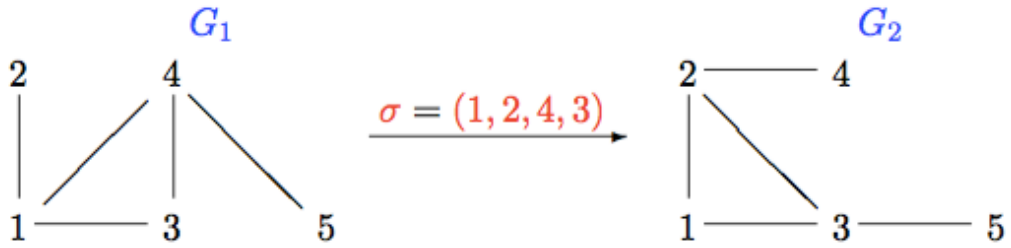

Figure 2.3: An example of challenge to solve in a zero knowledge proof protocol. Image courtesy of [Sma03].

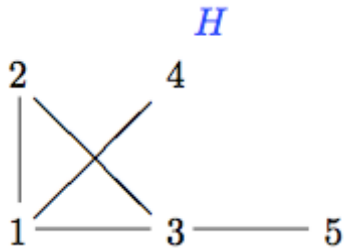

Figure 2.4: Peggy's committed graph. Image courtesy of [Sma03].

We refer to a ZKP as being complete if and only if, in case Peggy knows her solution is correct, Victor will definitely accept her proof. Moreover, a ZKP is considered sound if Victor has a small chance of accepting Peggy's proof when she is malicious, and has a false proof.

Chapter 3

Related Work

In this chapter we present related system and protocols that rely on cryptography in order to compute some value or statistical functions over a set of user location data. We analyse their initial consideration and approaches, in order to better defined our model's requirements. An introduction to k-anonymity and differential privacy is provided. We then end with the introduction of a framework for designing location-aware systems on which we are going to model $PrivMarket$.

3.1 VPriv

In [PBB09, Pop10], VPriv is introduced for drives that need to compute a cost function over their traveled path without revealing their journey. A path is a sequence of points, composed by a map position and a timestamp. The system is proposed for electronic tolling, delay and speed estimation, or pay-as-you-go insurance applications.

The participants of the VPriv system are drivers, their cars and a server. Drivers have a client application that can be either a mobile or web application, cars have a transponder that is able to infer the car's location and perform some communication with the driver's application and the server, and the server has logical and storage capabilities.

There are three main stages in the system, namely **registration**, **driving**, and **reconciliation**. These provide the protocol for computing the cost function for each driver in the system. VPriv also handles accountability of the users in the system - it provides the grounds for discovering misbehaviour on the drivers' side. We now expand on each stage of the first protocol:

3.1.1 Registration Phase

The client firstly requests permission for uploading a new set of path points, i.e. tuples of $< location, timestamp >$, representing the user's travel location points and their corresponding journey time). They are authorised a future upload by a **cryptographic commitment**(See 2.4.1.), for a set of anonymous ids, (or tags). What is more, the commitment scheme has a **homomorphic encryption property** (see section 2.4.2), but we will deal with this trait in the reconciliation phase, in subsection 3.1.5.

The client identifies themselves, by providing a licence or some other registration information. The client application then generates a set of random tags, $v_1, v_2, ..., v_n$ to be used later on during the driving phase. The tags are encrypted using a function f_k from a publicly known pseudorandom function family, by choosing k. According to [NR04], as cited by [PBB09], this collection of functions,

$f_k : D \longrightarrow D$, where D is the set of all possible values of the tags, has the following properties, given that one chooses k at random:

- $\forall v \in D$, $f_k(v)$ can be computed in polynomial time.
- f_k cannot be distinguished from a true randomizing function, for each input.

After encrypting the random tags with f_k, the driver **commits** to the values and to the function index k with respect to the encryption provided by the commitment scheme:

- The client application sends $c(k)$, the chosen function index k, encrypted.
- Then, it sends out the cipher texts of the encryption of each chosen tags, $c(f_k(v_i))$, where $i \in 1, ...n$.

The initial tags are encrypted in order to differentiate between cars that commit on the same anonymous tags. This is achieved by the choice of the pseudorandom function family. Lastly, the client application shares the unencrypted random tags with the transponder which uses them for uploading the location points. The client's application stores the function index and the tags, together with their **decommitment values** (see section 2.4.1), $d(k)$, and $d(v_i)$ $\forall v_i$, respectively.

3.1.2 Driving Phase

During the driving, the car's transponder tracks its location, and uploads tuples of form **id, location, timestamp**, where **id** is one of the random tags. On each upload, it provides a random tag that has not been used before. The transponder also stores this data to be accessed later on by the client application in case of a reconciliation phase, that is explained below.

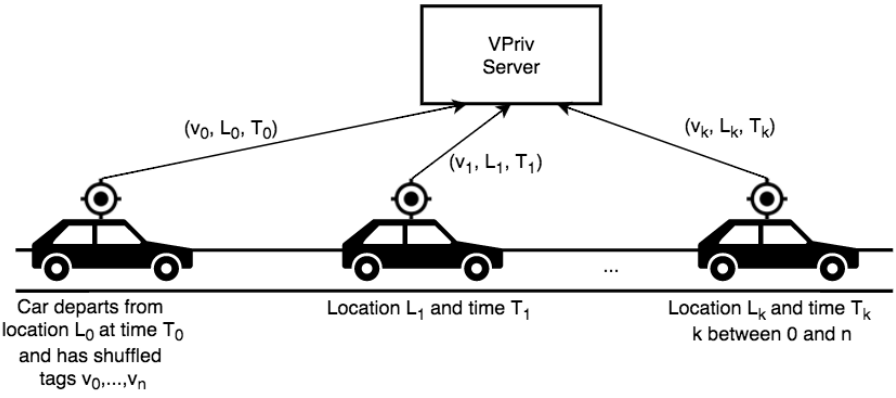

Figure 3.1: Driving Phase in VPriv: the user's transponder constantly produces location-time pairs and uploads them to the server using the anonymous tags.

3.1.3 Reconciliation Phase

At the end of a predetermined period, there is a reconciliation phase, where the user tolling is calculated. The driver uses the client application that operates on an authentication base. It checks if all transponder's data was uploaded, or anonymously sends out the unsent tuples. If the server requests a proof of accountability, the driver needs to prove that their uploaded path is consistent with the spot checks' uploads.

Then, the server computes the cost associated with each tag it received. The server does not know what tags belong to each driver, so it fetches all tags with a positive cost to the logged driver. The latter locally chooses their corresponding tags and performs the total. It sends the total value back to the server, and then it proves their calculation is correct.

Firstly, the user received a set of anonymous tags, together with their corresponding costs, and they choose the appropriate ones that matched their initial tags. Finally, they send the total cost back to the server and prove the correctness of the computation via an implementation of a zero-knowledge proof protocol (See 2.4.5.).

The proof takes place over multiple rounds in order to increase the **trust** of the server.

3.1.4 VPriv Path Protocol

The entire protocol is summerised below. Given a triple $< C, T, S >$, corresponding to the client's application, car's transponder, and the server, the system protocol can be describe as following:

1. **Registration**
   (a) C chooses a set of random tags $v_1, v_2, ..., v_n$, and a function f_k from a set of available random functions, by choosing k.
   (b) C encrypts each tag by the random function, $\forall v_i,\ f_k(v_i)$, and *commits* to these values by computing $c(k)$, and, $\forall v_i,\ c(f_k(v_i))$ and sending them to S.
2. **Driving Phase**
   (a) C sends the tags $v_1, v_2, ..., v_n$ to T that uses them when uploading to the server.
   (b) T generates pairs of $< location, timestamp >$, and uploads them together with a tag, using a new tag on every upload.
3. **Reconciliation**
   (a) S computes the associated tolling cost, t_j for each random tag s_j that it received in the last period, based on the location and time where it was observed and sends (s_j, t_j) to C, only if $t_j > 0$.
   (b) C computes the total tolling cost, $COST = \Sigma_{v_i = s_j} tj$ and sends it to the server.
   (c) C proves to S that COST is correct via a zero-knowledge proof protocol.

Figure 3.2: Operational steps of VPriv's protocol.

3.1.5 Zero-Knowledge Proof Procol in VPriv

VPriv implements an instance of the general zero-knowledge proof protocol. For an introduction and more details on this, please consult section 2.4.5. At this point in VPriv, both the system (S) and the client application (C) have access to all uploaded tags with positive costs. C needs to prove to S that it correctly computed the total sum (from step 3(b)), COST, by incrementally building on S's trust.

The protocol has a fixed number of rounds, and the client partitioned their committed tags by these rounds in the registration phase. We now present a simplified version of this protocol, with one round: C shuffles all pairs (s_j, t_j) received from the server in step 3(a) and commits for the challenge by sending:

- Encrypts all s_j with the initial pseudorandom function f_k such that is able to differentiate itself from other drivers that chose the same tags.
- A commitment of the values received, $c(t_j)$, whil storing the associated decommitments.

Now the server can either challenge the correctness of the previous step, or the correctness of the COST value, by randomly choosing a bit value, b:

- If $b = 0$, C will prove the correctness of their commitment. It sends k, the shuffled set (s_j, t_j), and provides $d(k)$ and all $d(t_j)$ in order for S to verify the previous stage.
- If $b = 1$, C will prove the correctness of the COST value. The homomorphic commitment gives the server the possibility to verify the sum by multiplying the intermediate costs associated with the driver. In this case the driver gives the decommitment values from the registration, $d(fk(v_i), \forall v_i$. Now S know what values $f_k(v_i)$ are associated with the driver, and can choose from the challenge the corresponding cost commitments, $c(t_j)$. It multiples them (due to the homomorphic encryption property of the commitment scheme). The decommitment key of this product will be Δ, the sum of all decommitment values of the associated costs, $d(t_j)$, where t_j was a COST used by the client. Now C sends Δ to S and the latter verifies if the total COST is the same.

3.1.6 Accountability Protocol

Anonymity introduces the problem of accountability, as there is no authority that can verify the correctness of the user data.

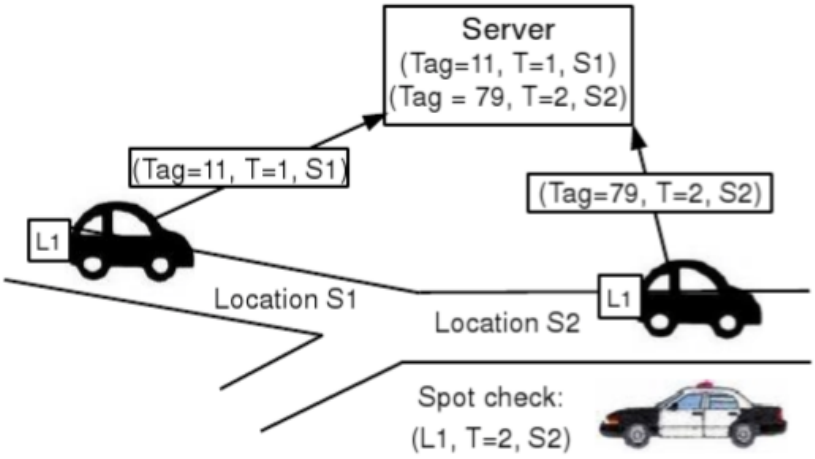

Figure 3.3: **User spot check**: A new entity, the checker, will record the user's licence plate number and upload it to the server together with the corresponding time and location. During reconciliation phase, the driver needs firstly to prove that his data was correct, before proceeding to calculate the toll.

There is also has an enforcement scheme in order to make sure non of the stakeholders misbehave. The driver's final cost correctness is given by the zero-knowledge proof in the reconciliation phase. However, it departs with the assumption that the paths points are correct and complete. Therefore, there has be a protocol to **make each driver accountable of the quality of the provided data**. VPriv threat model includes drivers' attempts of cheating by either turning off the transponder, spoofing its location, or pretending to be a different driver.

The accountability protocol consists of **sporadic random checks**. These are carried by control points that upload to the server the location-time pair of the observed car, together with its license plate. Then, in reconciliation, before computing the cost function. This is not achieved by any cryptographic mechanism, since the driver's license plate is known to the server for the observed location and time. The driver proves his correctness by presenting the tuple they uploaded for that point. A false tuple cannot be presented, because of the commitment scheme from the registration. The check is illustrated in figure 3.3.

3.2 Discussion on VPriv

During the reconciliation phase, the server might need to send large sets of tags for the client to choose from. This phase might imply huge communication costs, as well as storage costs on the client side. The suggested fix of this issue would be to cluster the map and attach prefixes to the anonymous tags such that the server fetches only a subset of the data set in order to allow the user to compute their cost. However, this might not provide sufficient correctness with a too finely-grained partitioning in practice, and the user might cheat if their journey goes beyond one cluster. At the same time, a too coarsely-grained partitioning might still impose high resource requirements on the user, and imply significantly unfeasible delays.

Also, during the zero-knowledge proof protocol, the server might challenge the client to prove the correctness of their commitment in which case they will learn some information of their path, by cross-referencing the inferred tags with the stored ones. [PBB09] provides a solution by polluting the database with dummy location-time pairs. However, this solution restricts the extension space of the system to other applications. On a random spot-check, the user will have to prove their uploads are correct, which might decrease the anonymity in a real-world system.

Lastly, this protocol is not suited for an aggregated function, where the cooperation of multiple users is required. Their cooperation is not feasible with the system's infrastructure and requirements.

3.3 PrivStats

PrivStats was introduced in [Pop10, PBBL11], in order to allow an **aggregator** (or collector, an entity that collects user data in order to do some computations over it) to compute some statistical (or aggregated) functions over the users' locations, while preserving the privacy of the latter. It was developed on the limitation of VPriv of computing statistical functions over the user data. The system applicability includes traffic studies, such as congestion metrics, traffic delays, or road statistics, like average speed, average number of travelers.

3.3.1 Overview of PrivStats

The system includes a set of users, a set of *smoothing modules* (SM) that flatten out user upload patterns, and a server that runs aggregated computations over the collected data. This overview is illustrated in the figure below.

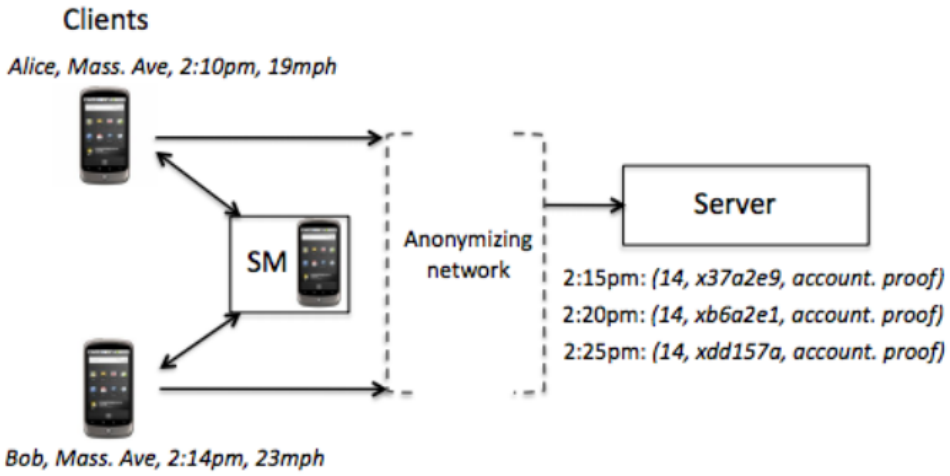

Figure 3.4: PrivStats Architecture Image courtesy of [PBBL11].

All the aggregates[1] are know in advance and the users willingly selects which aggregate want to contribute to, and then requests a participation quota. Each aggregate is publicly know, and it has an *id*, and a set of requirements - an area and a time interval of interest. All time intervals of the available queries end somt time in the future.

After subscribing to the interested queries to be computed, whenever the user meets the aggregate requirements (i.e. they are in the defined spatial area within the specified time interval), the upload tuples to the SM of the form (*id*, *sample*). The *sample* represents the individual's contribution to the query's goal. For instance, suppose there is an aggregate with $id = 3$ that has as requirements a road segment, S, and a time interval $[T_1; T2]$, with $T_1 \leq T_2$. Say the aggregate is interested in computing the average speed of the cars meeting the requirements. A client passing through S, in the time interval $[T_1, T_2]$, with available quota, will upload their speed value to the system. When the time interval of an aggregate has finished, the final query is computed. The collected samples are not useful anymore, since the tuples don't have any id or anonymous id attached to them.

A simplified overview of the underlying cryptographic blocks are outlined below. For an overview of the Paillier cryptosystems, please see section 2.4.2. We do not delve into more finely-grained details as they are not under the scope of this project.

1. **System setup** - SM generates a Paillier (PK; SK) pair.
2. **Client joins** - Client obtains the public key from the SM and subscribes to the desired aggregates.
3. **Upload Phase**
   (a) *Generation interval*
      i. Client generates a tuple $(id, sample)$ with the id of the subscribed aggregate and their requested sample data.
      ii. Client selects a random time t_s in the aggregate's specified time interval and does nothing until t_s.
   (b) *Synchronization interval*
      i. At time t_S, Client asks SM for s_i, the number of clients already engage in this aggregate
      ii. Client uploads δs, the number of tuple uploads is going to make to SM such that it preserves their anonymity and is consistent with their quota.
      iii. Client picks δs random times from the aggregate specified time interval in order to upload their tuples and uploads them to encrypted to the **Server**.
4. **Aggregation**
   - At the end of the aggregate time interval, the Server computes the statistical function using homomorphic encryption.
   - The Server asks the SM to decrypt the aggregated result.

Figure 3.5: Simplified running protocol of PrivStats.

[1]From this point onwards, we will interchangeably use *aggregated* and *statistical*, as well as *function* and *query* (since the tuples are store in a database structure, the computation of the function implies running a set of queries over the data). By *aggregates* we mean an aggregated query.

3.3.2 Accountability in PrivStats

In PrivStats no uploaded tuples is associated with any user identifier. Therefore, the system enforces accountability by having a sample point quota for each client's uploads, a total quota per client, and a check that the uploaded sample is within an acceptable interval. These are checked within a protocol based on a zero-knowledge proof. Due to time constraints, we only specify that this protocol is similar to $VPriv$'s.

3.3.3 Discussion on PrivStats

PrivStats aproaches the model design from a side-information attack perspective. A side-information refers to any information on any user from the system that was acquired out-of-bounds of the system. Thus, a side-information attack (on which we exapand later on, in section 4.5.1) aims to infer more information on the users from the system by using the latter.

The system also provides a scale for quantifying anonymity, by introducing plausible deniability and strict location privacy. We succinctly present them here, but we reiterate and expand on them in section 4.5.2, by adapting them to our model:

- **Plausible deniability** refers to the capacity of a user to deny they were at a particular location, regardless of the truth. It
- **Strict location privacy** refers to the capacity of the system of not revealing any information that might help an adversary expand their knowledge set on the system's users.

3.4 Enigma

In [ZNP15], a theoretical system is proposed for a peer-to-peer network that allows different entities to jointly store and run computations over data, while each party keeps their data private. It is a decentralised system that aims to serve as a platform for any general-purpose computations that preserve the privacy of data.

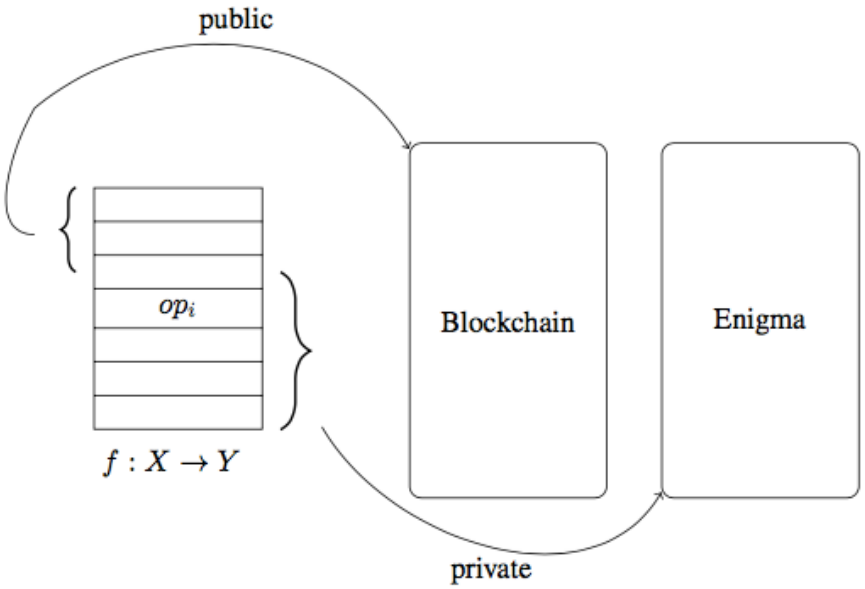

Figure 3.6: Overview of Enigma - there are two chains, a public one, the blockchain, and a private one, Enigma's. The access control policies, and other information that needs to be public is kept by the blockchain. Storage and heavy execution-based operations are processed by the off-chain. Image courtesy of [ZNP15].

It includes two components, an on-chain, the blockchain, and an off-chain network, Enigma. The first is a distributed ledger across the peer network with limited storage and computation capabilities, and is used for publicly keeping track of the correctness of the actions performed within Enigma. For more information on the blockchain, please refer to [Nak08], since it represents a departing point for potential future work, but it is not under the main scope of the project. The Enigma off-chain compensates for the blockchain's limitations:

- **Storage** Only the headers of the blockchain are stored in the distributed network, where each header has 80 bytes [Nak08]. Due to this limited storage capabilities, the Enigma off-chain has a distributed hash-table (DHT) with references on the location of the data, but not the data itself. The data is randomly split across the network, and no party has full access to the data. Now, the blockchain is used to store the access policies, that are made available to all parties, in order to check the correctness of each involved party.
- **Privacy Enforcing Techniques** Computations are based a on multi-party computation (MPC) scheme, where a third party is not needed to keep track of the correctness of the computation. Multiple nodes run the computation on their joint data sets, without leaking information to other parties. The correctness is guaranteed through a verifiable secret-sharing scheme (SSS), such as Shamir's Secret Sharing Scheme, presented in [Sha79].
- **Heavy Processing** The off-chain has the needed resources to perform heavy computations; therefore, the complex operations are done on it, and then the proof of work is broadcast through the blockchain.

The Enigma systems seems to bring a lot of advantages, such as privacy and scalability. It stand for a possible candidate when designing and modeling our system; however, its introductory document, [ZNP15] is a white paper - not enough details are offered in order to replicate the system in the time range of this project. The system is a theoretical proposition, but the authors suggest that a practical system will follow[2].

3.5 User Anonymity Paradigms

3.5.1 K-Anonymity

In [Swe02], the k-anonymity model is introduced in order to provide anonymity guarantees for a user within a data set. A collection of data achieves k-anonymity if each record in the set cannot be distinguishable from k-1 other records.

Formally, k-anonymity is a property of datasets of cardinality n ($n >= k$) that guarantees for any row, r_i, where $i \in 1, ..., n$ that there exists at least other $k-1$ similar rows, when the dataset runs a query on a subset of row's attributes.

3.5.2 Differential Privacy

Differential privacy [Dwo08] is a property that allows an entity to quantify the possibility of an adversary to infer further information on a single individual, given access to aggregated data by querying the corresponding database. Formally, this can be defined as following:

Definition 3.1. Let $\in$ be a positive real number. An algorithm M that takes as input a database and a query, is $\in$-*differentially private*, if for any two databases χ and χ' with $|\chi| = |\chi'|$, differing for only 1 record, and any query, q:

$$\frac{Pr[M(\chi, q) = r]}{Pr[M(\chi', q) = r]} < e^{\in}$$

[2]http://enigma.media.mit.edu/

In other words, the chance of the result of a query on being the same is almost the same, regardless on a changed input from a single user.

3.6 Location-Aware Frameworks

When dealing with location-aware systems that target to achieve location privacy, it is important to have a complete modeling framework. There are a number of theoretical frameworks that help one model a location-privacy aware system, such as [Dam14], or [SFH10]. For the general design approaches of PrivMarket, we have used first, while we kept the second only for its system evaluation approaches. We present its relevant part to the project when we do the threat modeling of our system, in section 6.2.

3.6.1 Design Framework

The framework relies on defining two underlying models that partition the semantics and operations of a system that entails user location data, an application model to be used in order to analyse the implied stakeholders, and the information flow between them. The privacy model is composed by three dimensions of its corresponding system under analysis, namely, its privacy goal, the used privacy mechanisms, and the privacy metrics. We will proceed in briefly describing each element of the framework.

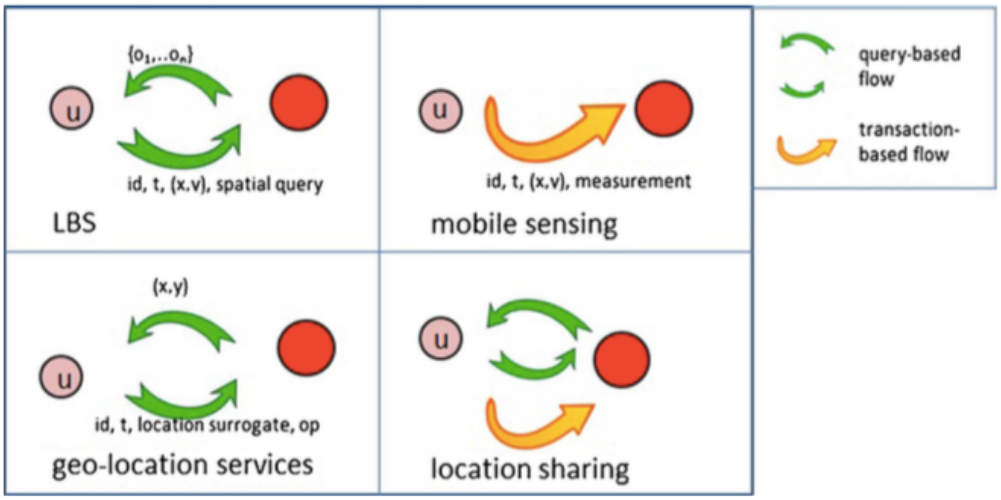

Figure 3.7: Information flows based on the type of single-party application, between a general user and a service provider. Figure courtesy of [Dam14].

1. **The application model**
   - Generally, a location-aware system is represented by a triple $(\mathcal{U}, \mathcal{S}, \mathcal{I})$, where:
     - $\mathcal{U}$ = the set of users in the system, where u is a general representative of the class.
     - $\mathcal{S} = \{s_1, ..., s_n\}$, the set of service providers; $|S| = 1$ implies a single-party application.
     - $\mathcal{I} = \{I_{s_1}, ..., I_{s_n}\}$, the set of information flows between users and services, where $\forall\ I_{s_i}$ represents the user flow between the service provider s_i and u, the general representative of the $\mathcal{U}$ class.
   - Any information flow element, I_{s_i} is characterized by:
     - *direction*, which can be query-based, or transaction-based.
     - *request content*, which is a tuple $< id, timestamp, location, content$, where id is an (anonymous) identifier of a user, and content can be empty, or application-based (such as a spatial query, a function, or a measurement).

  - *frequency*, which can be continuous (when the service requests take place close in time, and the user's location at time t_i is dependent on their previous location at t_{i-1}), or sporadic.
- Single-party applications can be location-based services, geo-location services, mobile sensing, or location-sharing services. (See the figure below for more information on these types of services).

2. **The privacy model** is an abstraction of a hard privacy solution (i.e., for a system that is not tru) that provides a partition on three dimensions:

   (a) The **privacy goal** can encompass one (or more) of the following: an identity protection (preserve the user's anonymous identity), a location protection (not reveal the exact user location, under any circumstance), as well as a behaviour protection (preserve the user's mobile patterns).

   (b) The **privacy mechanism** it generally consists of two functions. The first maps the user's id into an anonymous one, based on some criteria, while the latter maps the user's location onto a possibly different one. To be noted that the second one takes as input a pair of $(locationidentifier, timestamp)$ that the dermine the user's location, and can transpose it to a different pair $(locid', timestamp')$.

   Traditionally, there are a number of location function paradigms, out of which we mention location obfuscation (enlarge the area where a user is located), location perturbation (map the user's location to a different point, not necessarily near the initial location), location confusion (shift the user's location by introducing little noise in the original location), and location suppression (not providing the location data).

   (c) Finally, the **privacy metrics** are employed in order to quantify the location privacy guarantees. Each privacy metric should have a subject (single user vs. group of users), a stage at which is applied (before or after the observation period), and a paradigm employed (can be a k-anonymity underlying model, entropy-based metrics, error-based metrics, probabilistic-models-based metrics, or differential privacy).

Chapter 4

Design and Model

In this chapter, we provide a high level design of PrivMarket, a user data market. We then formally define the application model that delves us into the privacy model, the core part of our system design. In the latter, we analyse the possible threats and the limitations of the system, and we formally define the notion of location privacy with respect to the goal of PrivMarket.

4.1 The Challenge

We've presented in 3 formal systems that compute a function over user location data. Out of the presented systems, PrivStats seemed to be the best fit in the context of user data markets. However, as explained in 3.3, the system is designed with an aggregate-focus, that makes it unreasonably expensive in practice, mainly due to the high number of aggregates that might occur in practice. Clustering the predefined queries is not necessarily a solution, as it might leak information on the user's location.

Naturally, we ask ourselves: is it possible to leverage a data market approach, centered around the user's paths, as opposed to the aggregates? Can we put into use historical paths as well?

4.2 The Ideal User Location Data Market

Firstly, let us consider an ideal user location data market, that will be formally described in this section, defining the underlying notion requirements of such system.

Definition 4.1. User location data market (ULDM). A user location data market (ULDM) is defined as the tuple $< \mathcal{U}, \mathcal{A}, \mathcal{S} >$, where $\mathcal{U}$ is the set of users in the system (i.e. location data providers), $\mathcal{A}$ is the set of aggregators (i.e. entities issuing the statistical queries), and $\mathcal{S}$ is the aggregation point (i.e. the collection and logic entity) that responds to statistical queries on the data aggregated from the users.

In a real-world system that deals with privacy by anonymity, there is *side information* (SI), any information on the user that has an external provenience from the ULDM. This can be used by an adversary by cross referencing it with any data made available by the ULDM in order to find more information on the user (i.e. *side information attacks*). So, let there be an ideal ULDM, D, given by a tuple $< \mathcal{U}, \mathcal{A}, \mathcal{S} >$, where $\mathcal{S}$ can compute aggregated functions on historical paths of $\mathcal{U}$. Also, let there be a potential adversary, *Eve* that has access to side information (SI), via some methods - the harvest methods are described later on. Thus, we formally define our ideal ULDM in terms of side information:

Definition 4.2. Ideal user location data market (IULDM). An IULDM is an ULDM where an adversary, *Eve* can post an unlimited number of queries to the data market as part of a side-information attack and not learn anything new on the system's users.

4.3 High Level Design of PrivMarket

PrivMarket is a user data market that is composed by two core components - data collection and data querying. Each user in the network will record their path points, where a path point is defined as follows:

Definition 4.3. A path point is a tuple $(id, latitude, longitude, timestamp)$, where id represents an id that can prove the tuple's origin as valid, $latitude$ and $longitude$ represent a possible user location, and $timestamp$ is a date and time when the user was supposedly at the aforementioned location.

The id will represent a *cryptographic commitment* that the user is registered in the system. This protocol will be based on [Pop10] registration phase, and detailed in 5. Each tuple is stored encrypted in PrivMarket's main logic unit, where the data querying will take place.

During the querying phase, consumers (such as researches, advertisers, etc.) are able to study the statistics of the population's trends and flows. They pose statistical questions about the user network location in time, and PrivMarket will provide the result. The possible queries are presented in the privacy model, in 4.5.

4.4 Application Model

Firstly, there is a need of defining the modeling framework in order to structure our approach to designing and implementing the system. For this, two frameworks were used, as they have been specifically targeted for location aware systems, namely [1] and [2]. Thus, we will be preventing presenting the application model that serves as an introduction to the our system.

Let $PrivMarket$ be defined as the triple $< U, A, S >$. The modeled information flow of the system will be the following:

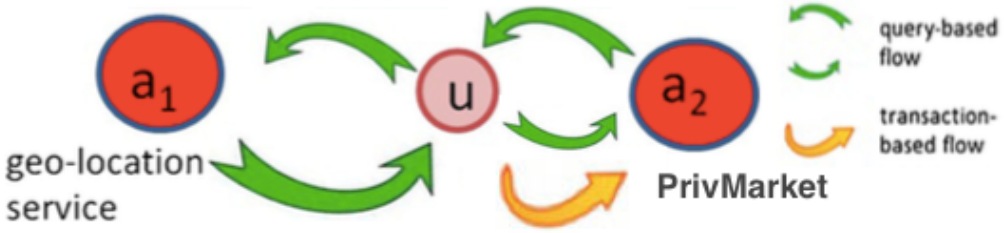

Figure 4.1: Information flow in $PrivMarket$.

The systems makes the following assumptions on the involved stakeholders:

- Users have smart phones with geo-positioning capabilities for locating themselves, either locally, or using a third party, such as a GPS system.
- The aggregators post queries about the user location data to the server.

As [DK06] mentions, there are three types of positioning system available to the user, namely *client-based*, *network-based*, and *network-assisted*.

- A client-based positioning system makes it possible for a user to locally compute their location, without revealing it to other parties. Examples here include GPS and inertial navigation.
- In a network-based positioning system, the user relies on the network infrastructure to compute their location, using the phone's GSM network. The position of the phone can be determined by using the identification information (called Cell Global Identity - CGI) of the cell that currently serves that particular phone. The administrator of the cell network will be in control of the computed location data as well.

- A network-assisted system is a hybrid of the previous two, combining both local data and CGI information. The administrator of the cell network might have a less precised information on the user's location.

While there is a trivial reasoning that the client-based positioning systems are desired to be used over the other two, the mobile user might not have enough power or storage resources for computing and storing, respectively their historic paths over a large period of time. However, the study of this information flow is not under the scope of the current project, and we will therefore assume one of the following will always hold:

- The user can use a client-based positioning system to compute their (latitude, longitude) pair, as accurate as possible.
- The user will use a trusted party for computing their location, on which we will only make soft-privacy considerations.

As illustrated in figure 4.1, there are 2 types of information flows between the system and the user, namely a transaction-based flow, under which the user updates their location data, and a query-based flow that is used for authentication. The latter can also be reused for implementing an incentive-based system.

4.5 Privacy Model

Following the generic location-aware model developed by [Dam14], in this section we present the privacy model of $PrivMarket$. Besides the security requirements imposed by such a system, the location data market comes by with a new first-class threat: side-information attacks.

We define the goal of location privacy for $PrivMarket$, in the presence of side-information attacks, and we further analyze the possible threats of the system. In this context, we derive the possible statistical functions that can be applied over the user network. For the used cryptographic material in this section, please refer to 2.4.

4.5.1 Side Information Attacks

Leaving from [PBBL11], we define side-information as below:

Definition 4.4. Side Information (SI). For any given ULDM, side information (SI) is any out-of-system-bounds information on any user, being obtained without direct or indirect usage of the ULDM.

In our case, a side-information can simply be generated by simply observing an user, say Alice, at a particular tube station. However, an adversary having ownership of such side information can lead to a side-information attack. Let there be U, the user set previously defined, Eve, and a user data market, D, that answers location-time input queries with statistical results of users whose time-location history satisfy the input requirements.

Definition 4.5. Side-information attack (SIA). A side-information attack (SIA) is a triple $< U, Eve, D >$, where Eve tries to further infer (new) information on any user in U, by using any SI previously acquired and by querying D.

There can be more types of side-information attacks, some of them being remarked in [Pop10, STLBH11] as well. We will define them in terms of $PrivMarket$ as follows, using users $Alice$ and Bob, and an adversary Eve for illustration purposes:

- **Areas of low density or popularity**. There might be areas from which there is a scarce location data. For instance, if $Alice$ is the only user living in a remote area, Eve can monitor $Alice$'s activity by querying the system for inbound or outbound travels with respect to that area.

- **Physical observation**. An adversary might observe a particular user at a specific time and location. For instance *Eve* observes *Alice* at a specific time and location. Even if the area is crowded at that point in time, if *Eve* knows that *Alice* is part of the system, she might achieve a *presence disclosure* attack. On querying the system, *Eve* knows there is a high probability of *Alice* participating in computing the aggregates corresponding to that area and time frame. **Presence and absence disclosure** can be dangerous to the user. For example, if *Eve* can infer when *Alice* is at home or not by querying the system, she can deduct the best time frame to break into *Alice*'s house.
- **Side information on other users**. *Eve* can make use of other user's SI in order to derive more path-related content. See the example below for an illustrated case.
- **Other**. There might be other side-information attacks based on various SI, such as map's topology (the roads, public transport network, etc.), commuting patterns, user time patterns (for instance, *Eve* knows *Alice* is always leaving home at 8:15), and others. These might lead to different types of attacks that are explored more in depth in the threat model in 6.2.

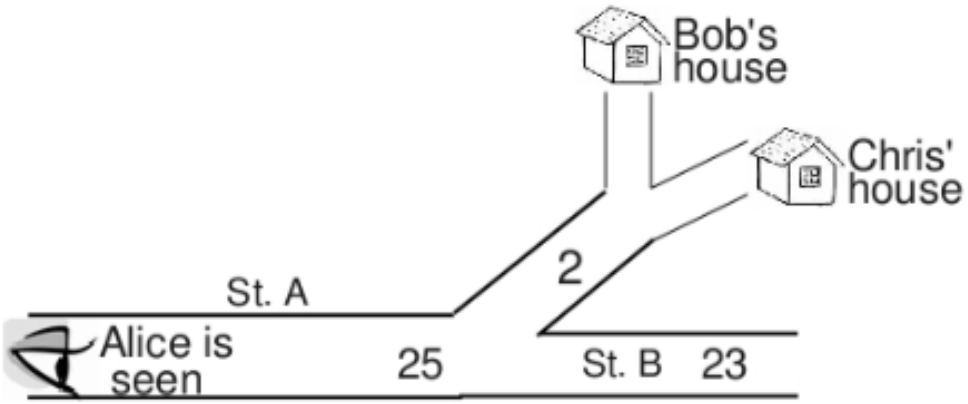

Figure 4.2: Example of side-information attack on a user using side-information on other users. *Eve* sees *Alice*, *Bob* and *Charlie* on street A and she knows where *Bob* and *Charlie* live. Thus, she can deduce that *Alice* went on street B. Image courtesy of [Pop10].

Since the queries are known *a posteriori* to the data collection, the system can now be targeted to a new set of side-information attacks, due to an adversary **knowing** the operational semantics of the data market system. Since the market runs the queries *after* collecting the data, we identify the following second set of side-information attacks:

- **Highly targeted queries**. *Eve* can try to pose queries with extremely finely-grained specifications that can only identify a small number of users.
- **Consecutive queries**. In this case, *Eve* can try to pose consecutive queries varying one dimension only (for instance, adding a Δt to time, or a Δl to latitude or longitude), and infer the difference between the two queries. The difference of results is similar to a SIA based on an area with low density of popularity, however the specification (location difference or time difference) can be extremely low, leading to the previous type of SIA.

In order to deal with this second set of SIA, more abstracted queries can be imposed by leveraging them over the semantics of the map. Shorki & al., [SFH10], introduce a 3-layer map model, composed by:

- **Geographical space**. This layer independently represents any point on the map. For this, we consider the latitude-longitude representation system for referring to points in this layer.
- **Location sites**. This adds the first layer of semantics, by applying the map topology (i.e. the road and common transport infrastructure), and building semantics (i.e. hospitals, schools,

houses, etc.).
- **Types of sites**. This last layer clusters the location sites based on their purpose, including, but not limiting to: shopping, cultural, sporting, residential, etc.

Based on this 3-layer view, *PrivStats* dissociates from the intial set of queries (that are called **syntactic queries**), and introduces **semantic queries** as an alternative, restricting the aggregators to commonly defined locations (such as stop names, street names, etc.); thus, the alternative corresponds to the second layer of the previously introduced map layering. The two types of queries are analysed further on in parallel. The third may represent a future work option to be taken into further study.

4.5.2 Privacy Goal

As opposed to PrivStats (3.3), PrivMarket introduces a new aggregation dimension, namely paths over time. Thus, we need a revisited scale for privacy, incrementally built with the goal of an IULDM, focusing on side-information attacks. Therefore, we define the following:

Definition 4.6. Plausible Deniability (PD). An ULDM has plausible deniability (PD) if and only if $\forall u \in \mathcal{U}$ that updated a path point to $\mathcal{S}$, u is able to credibly deny that was at that particular path point.

Now, let there be an ULDM that computes an aggregated result, R, based on a data set, D, and another aggregated result R' based on D', where $D \subset D'$. We also consider an adversary that has SI. Also, let there be any arbitrary u, $u \in \mathcal{U}$.

Definition 4.7. Strict location privacy (SLP). The ULDM is said to have strict location privacy (SLP) if and only if the probability that the adversary guesses u's path, given SI and R, is the same as the probability of guessing u's path when the adversary is given SI and R'.

Definition 4.8. Perfectly Strict location privacy (PSLP). The ULDM is said to have unbounded (or bounded, respectively) perfectly strict location privacy (unbounded PSLP, or bounded PSLP respectively) if and only if, given an adversary, *Eve*, where $Eve \in \mathcal{A}$, with an unbounded (or bounded, respectively) set of correctly issued queries to $\mathcal{S}$, for each obtained result, R, the ULDM achieves strict location privacy.

Now we can formally define the privacy goal of our $PrivMarket$, as **achieving PSLP in the presence of side-information**.

4.5.3 System Model

Since $PrivMarket$ aims to run statistical queries over past location traces, it is reasonable to consider a storage layer in the system design. Our theoretical choice of a storage layer is a homomorphic encryption scheme. However, FHE is prohibitively expensive, and we need to search for *specialised encryption schemes* - the overall design approach is outlined in the figure below.

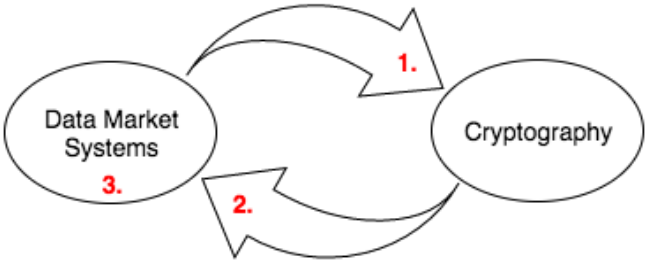

Figure 4.3: Our approach is given by three steps: 1. understand the underlying requirements of the model; 2. find specialised encryptrion schemes; 3. integrate them and build the system.

Assuming we have such a storage layer, the overall view of the system is pictured below. Each user needs to be able to upload their location, and each consumer (or aggregator) needs to be able to run a statistical function, without being able to infer more about the data.

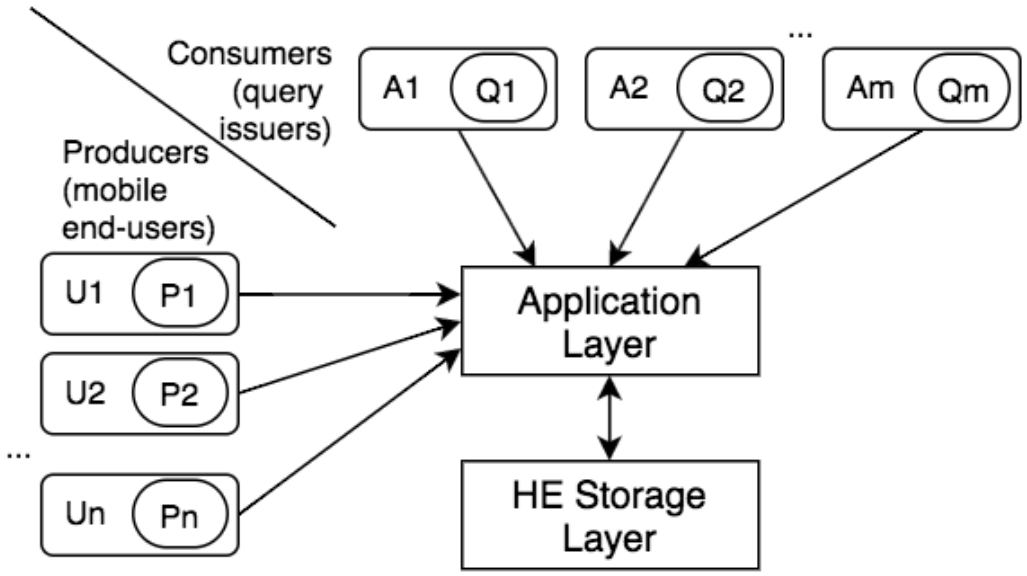

Figure 4.4

The application includes the logic for running a data market. In an actual implementation, this component might be broken down into multiples one. For example, one possible This layer handles for each stakeholder:

1. **Producers:**
   (a) **Registration** can be based on a commitment scheme, similar to the $VPriv$ system presented in subsection 3.1. Given a family of pseudorandom functions, each user randomly chooses a function from this family set and a set of tags and *commits* on these values to the server. The registration process is not covered in the proof of concept, and, due to time constraints will not be included under the scope of the project. We also assume that the users assume an anonymising network (such as Tor) to hide their identity.
   (b) **Data upload** for each user is tuple of $(id, location, timestap)$, where id is one of the previous randomly chosen tags. The same tag will be used for one journey, since we are interested in computing the functions over paths. The definition of the journey is based on the market application. For instance, in our proof of concept, a journey represents one travel via the London transportation network.
   (c) An **accountability check** module needs to be placed in order to detect any misbehaviour on the users' side. Developing and implementing a general protocol is not subject of this project, due to its time limitations. However, we provide a solution for accountability in our case study.
2. **Consumers:**
   (a) **Registration** for consumers does not need to be private. They go through a thoroughly registration process in order to offer users' control over the set of entities that use their data. Thus, each physical person running aggregates should be an employee or a representative of a company, research group, or authority and will identify themselves upon registration.
   (b) **Aggregate computation** starts when the consumers issue the desired aggregation function and the application layer checks that it does not violate any thresholds of the system that might lead to a side-information attack.

(c) **Accountability Check** is not included in our project, but proposed as a future investigation point. We just note that in a centralised data market, the misbehaviour of the consumers can be checked at the application level. Since they have identified themselves upon registration, a penalty system can be implemented.

4.5.4 Aggregate Queries

Given the time constraints of the project, we restrict the scope of the project to basic aggregated statistics based or derived from count functions. We then provide a discussion on what further queries can be developed.

Table 4.1: Possible queries run in $PrivMarket$

No.	Type of Query	Examples of Queries
1.	Counts over 1 area	How many users were in area X in the time interval T? How many users entered (or exited, respectively) area X over time interval T? How many users passed through area X over time interval T?
2.	Counts over 2 area	How many users traveled between area X and Y during time frame T?
3.	Averages over long periods	What is the average number of people passing through X over a week?

This functions are pictured in the figure below. They can later on be expanded to other statistics, such as more averaged statistics (for example, average journey time), standard deviation, or other application-based statistics that take as input some application-defined content joined with the location paths, over which the statistics are run.

However, for the purpose of the project, we will study how the basic block queries influence our design, together with their strengths and limitations. In section 5.3.2 we provide the examples particularized to our case study.

Chapter 5

Implementation

As mentioned in section 1.3, specialised encryption schemes come in as fast alternatives to fully homomorphic encryption which is prohibitively expensive. Given the goal of our data market, to perfom computations over **historical traces**, it was sensible to look for storage options. In this chapter we present a case of study on which we base our proof of concept, and we provide an analysis of CryptDB, the underlying technology for running computations over a data set. We provide a basic implementation of the case study that we integrate with CryptDB. The choice was based on the advantages CryptDB offers, such as SQL-based encryption and adjustable query-based encryption, that we present in the following subsections.

5.1 Introducing the Case Study: Transport of London (TFL)

Most of the location data sets are not made publicly available for research purposes. Even location data collected as part of research studies is usually obfuscated up to the point of limiting its usability. We initially considered the data collected by Device Analyzer, presented in [WRB14]. There was sufficient information on the location of the antennas serving the users - each antenna served the users within a predefined range. While there was the option of translating this data set into a theoretical graph nodes, based on the users paths, we considered to be more sensible to use an artificial data set, in order have more control over the quality and quantity of data to be used. Also, this was more relevant due to the technological advancement of mobile phones technologies t render their location at a more precised level, either by using GPS, or querying the surrounding network of GSM, or Wi-Fi providers.

We therefore propose a possible application of a data market, based on the Transport of London[1] (TFL) network. We assume that TFL is interested in studying the daily journeys of the people using common transport in London. There can be multiple benefits that can come from this, but we will classify them as a smart city approach, where the authorities would like to improve the transportation network based on a new set of statistics on their users. For the purposes of our proof of concept, a set of data was need in order to run the aggregates. Therefore, we used the Transport of London (TFL) API[2] to emulate the journeys of a number of travelers within a time frame. More information on the generation of the data set is provided in subsection 5.3.1. For the initial setup of the case study, we built the unencrypted data set based on the following considerations:

- Every user had a fixed departure point, symbolising their 'home'.
- 70% of users had regular traveling paths during the weekdays (emulating their commute), and random paths during the weekends; the rest had completely random travels within the week, in order to query both travel patterns and seemingly random journeys.
- 50% of all users have daily shifts, simulating a morning-evening commute, while 20% of all users simulated an evening-morning commute.

[1] www.tfl.gov.uk
[2] https://api.tfl.gov.uk

- A journey classifies as a morning commute if it is in the time interval $[6am; 12pm)$, while an evening commute is in the time interval $[4pm; 22pm)$; random journeys can take place anytime in $[7am; 23pm)$.

5.2 CryptDB

CryptDB offers such specialised encryption schemes in order to execute SQL queries over encrypted data. It can be used by database-backed services that have the application and the database components decoupled on different server instances.

5.2.1 Overview of CryptDB

CryptDB has two components, a database proxy and a DBMS extension for running computations over encrypted data. The latter goes on top of a regular database management system (DBMS) and uses the proxy server to handle all requests (i.e. queries) addressed to the database. The proxy receives the queries from the application server, encrypts them, and sends the encrypted queries to the DBMS server that executes it and fetches an encrypted response back to the proxy. Finally, the proxy decrypts it and sends it back to the application.

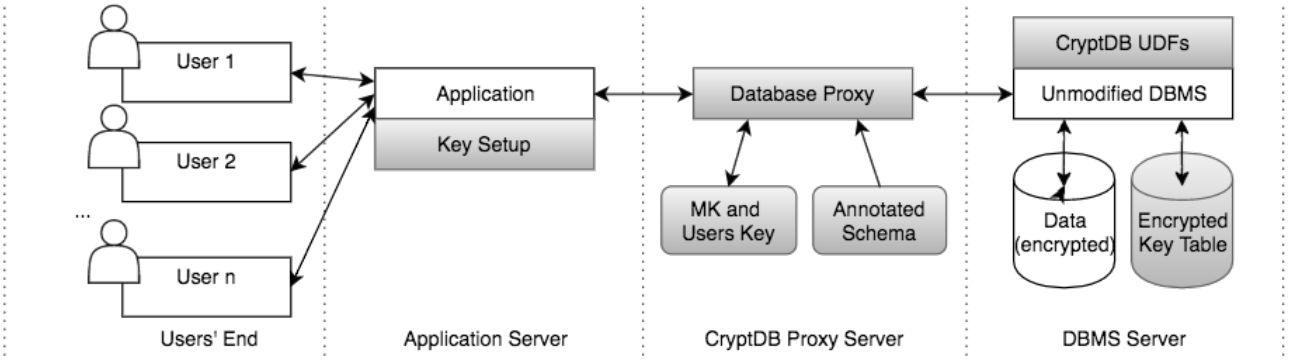

Figure 5.1: CryptDB's architecture consists of two parts, a proxy server and an extended DBMS that uses user-defined functions (UDFs) in order to perform computations over encrypted tables.

All the data inserted into the database is encrypted over multiple layers (See subsection 5.2.3, and the table and column names are anonymised by the proxy. The DBMS server has never access to the unencrypted data. The CryptDB DBMS layer provides user-defined functions that are used to perform computations over the encrypted tables, depending on the type of data in each column. Figure 5.1. presents the overall architecture of CryptDB and how it integrates in a possible multi-user application system.

5.2.2 Journey of a Query

The proxy has a master key (MK) that is used for encrypting the queries. It also has the database schema, and the encryption level of every column in the database. When the application issues a query to the proxy, it triggers the following flow:

1. The proxy rewrites the the query, by anonymising the names of the tables and the columns and by using the MK to encrypt each constant in the query with respect to the corresponding column and the specified operation. (section 5.2.3).
2. The proxy checks if the DBMS should adjust encryption level of the data in order to execute the encrypted query (section 5.2.4).

3. The proxy sends the encrypted query to the database that **runs the query over the encrypted database** and computes the result of the query.
4. The DBMS sends back the encrypted result, and the proxy server decrypts it and sends the result back to the application.

5.2.3 Onion encryption layers

In this section, we refer as *plaintext* to any unencrypted value (that can be an integer, a string, etc.) that will be inserted or used in the database, and as *ciphertext* to its encrypted equivalent, whenever we want to abstract away the type of the value. The CryptDB proxy pipelines each plain text through multiple encryption onions, in order to model their security with respect to utility. Each column has its own encryption onion, composed by multiple layers.

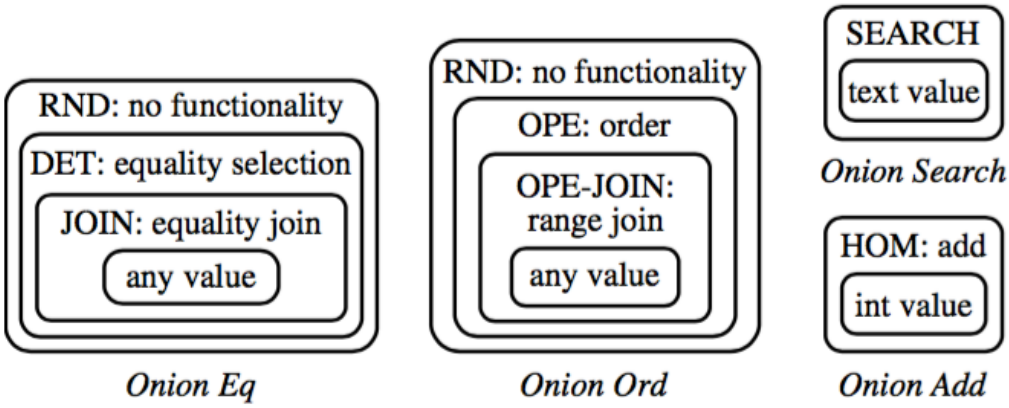

Figure 5.2: The onion layers for each supperted operation. The utility of the encrypted values is inversely proportionate with their security. Going inwards, utility increases; going outwards, security increases.

Table 5.1: The available SQL operators that can be performed by each layer of encryption.

No.	Layer of encryption	Supported SQL
1.	RND	No computations supported.
2.	DET	GROUP BY COUNT DISTINCT
3.	OPE	ORDER BY MIN MAX SORT
4.	HOM	SUM
5.	JOIN and OPE-JOIN	JOIN
6.	SEARCH	LIKE

The onion layers are applied on a column basis, independently of their table. Going inwards, the inner layers offer more utility over the cipher text, but sacrifice some anonymity of the data. Going outwards, more security is achieved, but the set of utility functions decreases. An encryption onion is built for each column (with respect to their type), for each supported operation. Figure 5.2 presents the initial state of all types currently supported in CryptDB. We now describe them individually, but provide an overview of the supported SQL operators in table 5.1.

Random Encryption

Each value is randomly encrypted. Since it is a probabilistically Two equal plain text will most likely be encrypted to different values. The scheme provides the highest level of data security, but no utility over the encrypted cipher texts - RND does not support any operations.

Deterministic Encryption

In a deterministic encryption scheme, two equal plain texts will be encrypted to the same value. Therefore, a database encrypted with DET leaks the equality property over the the column values. It supports equality-based operations: GROUP BY, COUNT, DISTINCT.

Order-preserving Encryption

For any two ordable plain texts, x and y, if $x < y$ then $OPE_k(x) < OPE_k(y)$, $\forall k$ in the encryption key set. OPE is a weaker encryption scheme than DET since it reveals the order of the elements in the collection. As [PRZB11] mentions, CryptDB uses [BCLO09] giving provable security to OPE - OPE is equivalent to a random encryption that preserves order.

Homomorphic Encryption (HOM)

As presented in subsection 2.4.2, homomorphic encryption is a secure probabilistic scheme on which one can perform computations over the encrypted data.

$$\forall \text{ plain texts } m_1, \text{ and } m_2, \text{ we have that there exist two operations, } \diamond \text{ and } \circ \text{ such that}$$
$$E_k(m_1 \diamond m_2) = E_k(m_1) \circ E_k(m_2), \text{ for any encryption key of the scheme, } k.$$

CryptDB uses the additive homomorphic encryption of the Paillier cryptosystems (subsection 2.4.2), where $\diamond$ is the addition operator, and $\circ$ is the multiplication operator. Thus, HOM covers **SUM** queries. Whenever the DBMS needs to perform a summation, it uses UDF mapping where a multiplication function overrides the initial summation one.

HOM might be able to support average queries as well, by having the CryptDB proxy do two queries - a SUM and an COUNT of the needed elements. The proxy performs the final operation locally, after decrypting the results, since the added overhead is negligible, assumint the proxy sends numerous queries to the database.

Join (DET-JOIN and OPE-JOIN)

There are two types of join queries, namely *equi-joins* (based on equality, using the $=$, IN, or $LIKE$) and *range-joins* (based on ordinality, using the SQL constructs $<, \leq, >, \geq$, or $BETWEEN$). Since the onion layers are applied on a column basis, one cannot perform an equi-join (or a range-join, respectively) between two columns (assuming the same type) that are encrypted in the DET (or OPE, respectively) layer because they will have different keys. Applying the same onion layer key in advance on all columns which can potentially support a join decreases the security level of the database. In that case, the actually performed set of join queries becomes a subset of the supported one, a poor security design choice, especially if we can dynamically adapt our encryption schemes.

Therefore, new different schemes are requested that lower the security level of data on the fly, when a join query is requested. If the JOIN queries to be computed would be known in advance, the CryptDB proxy will know when to reuse a DET or OPE key. If the joins are known afterwards, CryptDB differentiates between the equi- and the range-joins.

The first is handled by an intermediate encryption scheme, *join-adjustable* (JOIN-ADJ). JOIN-ADJ is a deterministic key-based hash function (so non-invertible) with the property that the used key can be changed, by altering the output, **without accessing the plain text**. This encryption scheme is based on elliptic-curve cryptography. An elliptic curve is essentially a plane algebraic curve defined by an equation of the form $y^2 = x^3 + ax + b$, and $a, b \in \mathcal{R}$. For more information on this, see [Mil85]. The regular JOIN scheme is implemented as $JOIN(v) = JOIN - ADJ(v)||DET(v)$. Since JOIN-ADJ is non-invertible, the proxy can recover the initial value by decrypting the DET component. Given a join on two columns, c and c', their JOIN-ADJ hash keys, k and k', and a value v that resides in both columns, the proxy computes $\delta k = k/k'$ that has the following property:

$$(JOIN - ADJ'_k(v))^{\delta k} = JOIN - ADJ_k(v)$$

The server receives the value of δk and adjusts the value of JOIN-ADJ of column c' and then performs the join by using the JOIN-ADJ component. The DET components will still use different keys. Even though now the DBMS knows the relationship between the 2 columns, this does not affect the columns on which no JOIN has been performed. Also, the plain texts are still encrypted in the DBMS.

For an OPE-JOIN, the same approach cannot be followed, due to differences in the OPE scheme. Thus, the OPE-JOINS need to be specified, or let the proxy use the same encryption key in the OPE-JOIN layer. However, [PRZB11] mentions that the usage of such queries is relatively low; thus, excluding the support of this scheme does not affect most of the applications it can support. This can be applied to PrivMarket as well.

Word Search

For strings, CryptDB gives the possibility of running SEARCH queries. The text in these columns is split by standard delimiters (or other user-specified characters), a random permutation is applied on the tokens, and then each word is encrypted separately, using [SWP00], as [PRZB11] mentions. The tokens are padded during encryption, making the overall scheme close to RND. However, the SEARCH queries can leak the number of words in the LIKE clause, and regular expressions are not supported.

5.2.4 Adjustable Query-Based Encryption

Upon a column creation, it starts with RND as the out-most layer. However, the level can decrease over time in order to allow more operations to be run on the encrypted data. For instance, consider the following example:

Employees

ID	*Name*
23	Alice

Table1

C1-IV	*C1-Eq*	*C1-Ord*	*C1-Add*	*C2-IV*	*C2-Eq*	*C2-Ord*	*C2-Search*
x27c3	x2b82	xcb94	xc2e4	x8a13	xd1e3	x7eb1	x29b0

Figure 5.3: Left: Example of a plain table CryptDB needs to store. Right: The plain table's equivalent that gets stored by the DBMS after the proxy encrypts each column for each supported operation.

Let there be the right table above with two columns 'ID' of type integer, and 'Name', of type string. Initially each column is encrypted up to RND, as previously mentioned. Then, the DBMS will store the equivalent table from the left. Assume that we would like to fetch all rows that have the ID cell 23. The SQL query that arrives at the proxy will be:

SELECT Name FROM Employees WHERE ID=23;

The proxy checks the minimum needed out-most layer in order to compute 'WHERE ID=23' on the first column - it should be DET, to support the equality operation. The current layering for equality is

$RND \longrightarrow DET \longrightarrow JOIN - DET$, with RND as the out-most one (i.e. the one seen by the DBMS); therefore, the proxy gives the DBMS the key to decrypt RND, updating the encryption layering to $DET \longrightarrow JOIN - DET$:

UPDATE Table1 SET C1-Eq = DECRYPT_RND(K_T1,C1,Eq,RND, C1-Eq, C1-IV);

The corresponding encryption key of the RND of column 1 is K_T1,C1,Eq,RND. Assuming that in the previous result, we got that $C1 - Eq = x7a12$, the proxy sends the next request:

SELECT C2-Eq, C2-IV FROM Table1 WHERE C1-Eq = x7a12;

Both $C2 - Eq$ and $C2 - IV$ are fetched such that the proxy can decrypt the value of $C2 - Eq$ and get 'Alice'. At this point, if the proxy issues another query that needs to run equality over the 'ID' column (for example, SELECT * FROM Employees WHERE Name='Bob';), the proxy doesn't need to reconfigure the current encryption layer of the column 'ID', and just asks the DBMS for the currently stored value (and the IV).

5.2.5 Choice Discussion

We chose to use CryptDB for our implementation due to its the previously presented features of its specialised encryption schemes. In its general configuration, CryptDB assumes that the Proxy is secure. If integrated with PrivMarket, this might not be a sensible assumption to make. However, we consider this as a departure point in building PrivMarket. Our proof of concept is based on these considerations. We also discuss another feature of CryptDB, key chaining that extends the initial model with a proxy that is susceptible to attacks. Based on this, we then talk about a possible improvement of our first solution.

5.3 System Implementation

We follow the general implementation scheme of CryptDB, as suggested by [PRZB11]. The producers (mobile users) are uploading their location points to the server. A location point is a tuple **(id, latitude, longitude, timestamp)**, based on a current or previous location where the user has been. The consumers are only allowed to run aggregated queries. Both parties interact with the system in the following way, as pictured below.

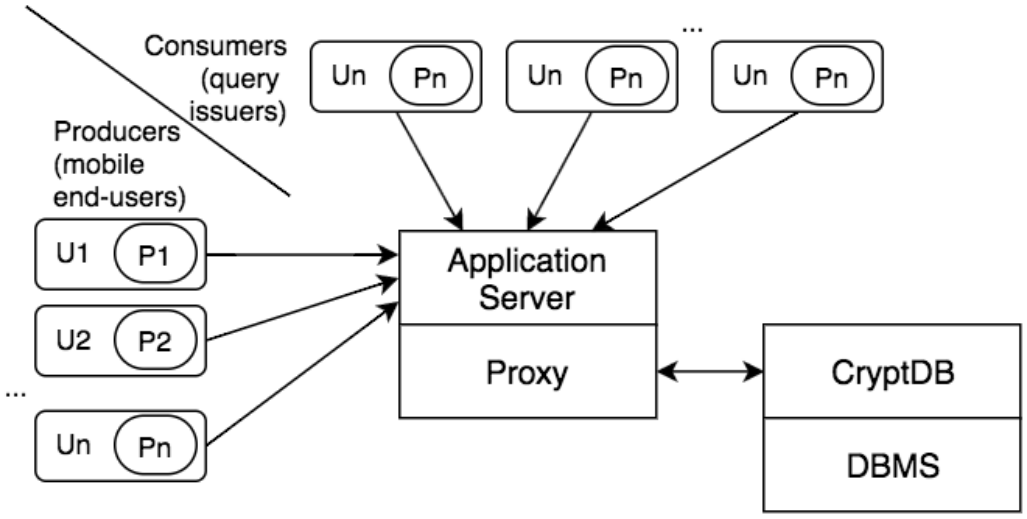

Figure 5.4: System components and stakeholders.

5.3.1 Aggregating the User Data

In our proof of concept, we simulated the traveling activity of **100,000 users** over a period of **14 days**. We assumed that each user uploads their locations while traveling. Although this assumption might not be feasible, especially given the London tube network, we used this topology by querying TFL API on the schedule, and any live updates.

Each user was assigned a constant departure point, to simulate their **home**. We consider two types of travelers - commuters, that can either work on a morning or night shift, and random travelers, that always travel to a different destination point. For the purpose of this proof of concept, we assume only one journey per user, per day. Over the weekends, we assumed all travelers are random.

For each user, we firstly computed their home, departure and arrival time, and, if applicable, their commute point for the weekdays. The 'home' and outbound stop points are randomly chosen from the set of all stops of TFL tube, overground, or DLR. The type of commuters imply three time intervals for starting a new journey, we set them to be: *morning time intervals* between 6am and 12pm, *evening time intervals* between 4pm and 22pm, and *random time intervals* anytime between 6am and 23pm. Then, for each day in the defined time interval, between 1ŝt and 14t̂h May, exclusively, we queried TFL for each users, simulating they used the traveling path provided by TFL. The table below gives an overview of our weekday considerations:

Table 5.2: Traveling times of simulated users in $PrivMarket$. Here *both* means outbound and inbound travels.

No.	Type	Weekend travelling			Weekday travelling
		Day [6am - 12pm)	Evening [4pm; 22pm)	Any time [6; 23pm)	Any time [7;22pm)
1.	Morning	outbound	inbound		both
2.	Evening	inbound	outbound		both
3.	Random			both	both

5.3.2 TFL Queries

For this proof of concept, we initially had the queries on an area coordinates. Without losing our generality, we assumed rectangular areas, where the North-East and the South-West points implied the area of the desired rectangular. Other easy-to-be calculated shape would have fit our requirements. Then, the user would prompt their desired area by inserting the coordinates of this area, when plotting the latitude and longitude values on a two-axis system, as pictured below.

As presented in subsection 4.5.1, side-information attacks can allow an adversary infer more details on the system's users. Based on [SFH10], we presented in 4.5.1 three types of map layers: geographical space, location sites, and types of sites. With these in mind, we propose **semantic queries** in order to restrict a potential adversary from performing a side-attack that might infer, for example, the presence of a person in a building. Thus, we propose the following two classes of queries:

- **Syntactic queries** - the *WHERE* clauses in the queries are based on latitude-longitude coordinates like presented so far.
- **Semantic queries** - the *WHERE* clauses have predefined values, based on the map's topology (road infrastructure, buildings, etc.)

The power set of all semantic queries will represent the set of all possible queries that can be run on the system. Therefore, from the location perspective, the queries will be known *a priori* to the computations. The time interval values will remain the only continuous random variable, from the computationally bound perspective of the system.

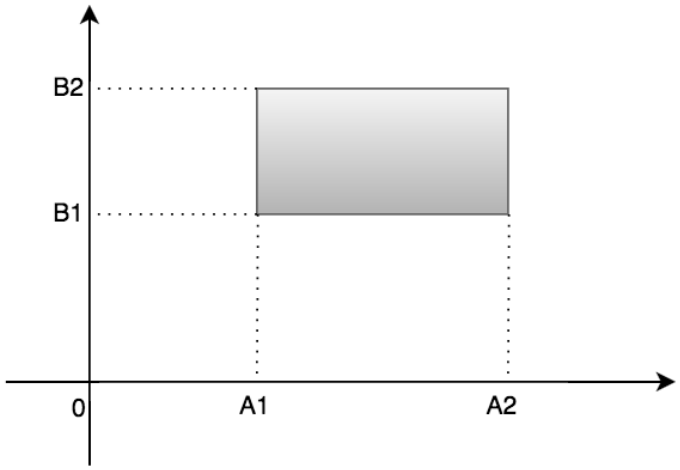

Figure 5.5: We consider the cartesian axis system mapped on latitude-longitude degrees. Then, each rectangular area will be defined by 4 values - A1, A2, and B1, B2, respectively.

In the proof of concept we run the following types of queries:

- **Count the number of users in area X, given the time interval T**. This takes a snapshot of the users that already reported themselves as being in area X, and adds any further users entering the area. Alternatively, the query can be altered not to take into account the initial subset of users (count the number of users entering area X during the time interval T). Similarly for an 'exit' case.
  If a users enters the area multiple times, it will be counted on a day basis. If, for instance the query is for a period of a week, and in the specified area *Alice* commutes during the weekdays, she will be counted 5 times.

- **Count the number of users in station X, given the time interval T**. The semantic equivalent of the query above. For the proof of concept, the semantic topology of the map is composed only by the stations. The underlying implementation is still based on the previous type of query due to the limitations of CryptDB for strings.

- **Count the number of users traveling between two areas, given time interval T.** The areas are defined similarly to the first type of query. The first represents the departing area, while the second is the arrival. In case they overall, the query defaults to a simple count of one area. We have chosen this implementation option, due to delegation reasons - it should be within the consumer's responsibility of having inserted the queries correctly in order to receive the desired computation.

- **Count the number of travelers from station X to station Y.** Similarly as above, we provide an overlay of abstraction above both the departure and the arrive location. X and Y are tube stations in the interface, but an internal hash map of the actual coordinates is kept in the back-end.

- **Average counts over long periods**. If interested in time ranges bigger than one day, we cover the aggregates above, but averaged over a longer time period. This includes, the average number of TFL travelers to a station in a week, or the average number of morning commuters between 2 stops. Now the utility of these queries gets shaped better when we increase the complexity of the queries.

The only time-constraints in the aggregation query in the proof of concept is given by the number of days of collection in the data set. To be also noted that in the case of an average request, the proxy

does two underlying queries, namely a SUM and a COUNT, given some input. This prevents the DBMS from learning more about the stored data.

5.3.3 Technologies Used

We now present the technologies used in the proof of concept, excluding CryptDB whose theoretical model was previously presented. The practical implementation that was used for CryptDB can be found on [pop].

Back-End

The back-end was implemented in Python, using Flask[3], a micro-framework that integrates another two frameworks: Jinja2, and Werkzeug. The first is a template engine, offering a clear template management and rendering system. On top of that, it provides a sensible solution for transferring JSON objects from the back-end. The second interface is useful since it provides a simple Web Server Gateway Interface (WSGI) toolkit that was used for handling all HTTP requests to the application.

Front-End

The front-end relies on Javascript for handling the back-end query replies, and for offering a more intuitive way of querying the database, due to the exploration purposes of the project.

The front-end also uses Google Maps API[4] that provides customization utilities, such as adding points, drawing areas and other ways of annotating maps, on any requested Google Map object. The rest of the used tool and technologies for the front-end are regarded as common and just enumerated: Bootstrap[5], HTML and CSS.

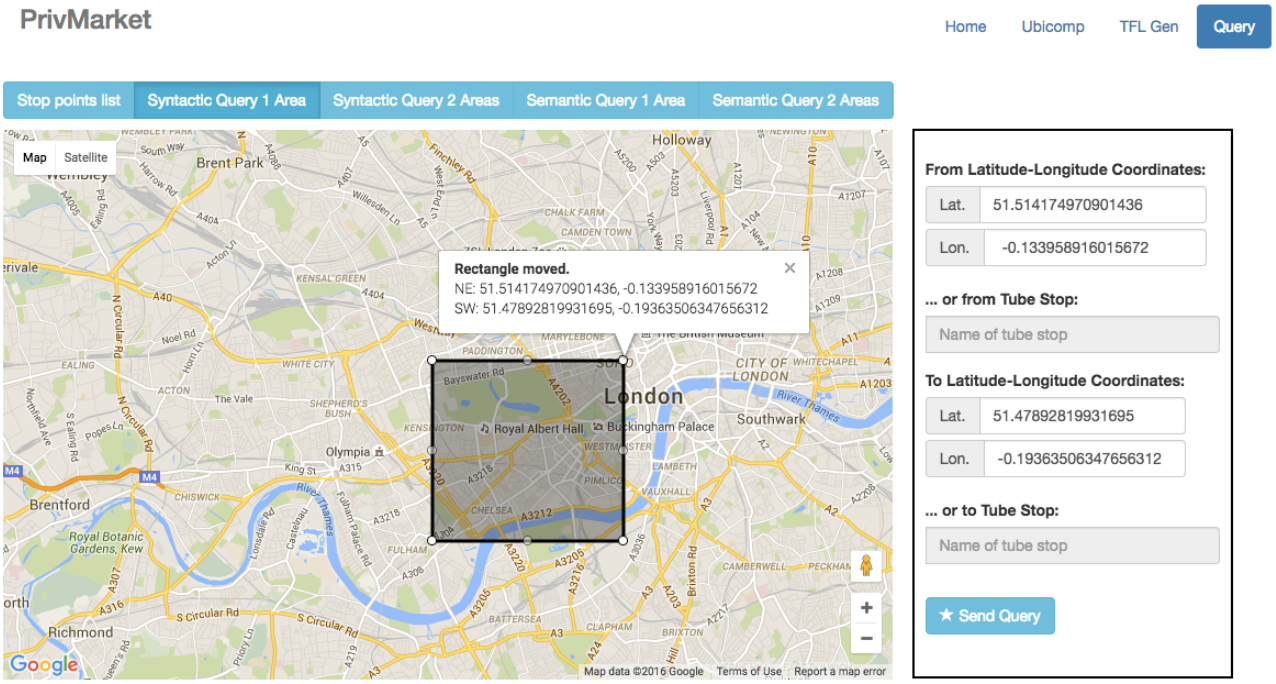

Figure 5.6: Query view for one area: the consumer drags and resizes the area as needed. On every map change, the form is updated accordingly. When the user is satisfied with the covered area, they can send the query and the result will be displayed.

[3] www.flask.pocoo.org/
[4] www.developers.google.com/maps/
[5] www.getbootstrap.com/

Chapter 6

Evaluation

We begin this chapter with a discussion on the limitations of our proof of concept, together with an overview of CryptDB's restrictions. Expanding on CryptDB's key chaining feature, we provide details on how the limitation of the first solution can be addressed. Finally we mentioned final remarks on anonymity and privacy of *PrivMarket*.

6.1 Limitations of Implementation

In a regular CryptDB application, there is no difference between the producers and the consumers. CryptDB was built primarily to defend the application's data from a curious, but honest database administrator that will try to infer as much as possible data on the system's users. In that model, the **master key** from which all encryption keys are generated, is kept with the proxy, that is assumed to be *secure*.

However, in *PrivMarket*, there is a distinction between producers and consumers; the first should be able to only insert encrypted data into the database, while the second needs to be allowed only to perform aggregated queries on the data, not necessarily in its entirety. Therefore, this difference needs to be reflected in the access control policies enforced by the system. CryptDB offers key chaining from which we can start sketching our solution.

6.1.1 CryptDB Key Chaining

As [PRZB11] suggsests, CyrptDB was built with two threats in mind:

- An honest but curious database administrator that behaves correctly, but is interested in inferring more information on the database.
- A possible attack that compromises both the DBMS and the application layer.

In case of the first one, we've seen homomorphic encryption provides a relatively reliable solution to the threat, within the scope of the supported queries. We discuss in subsection 7.2 a possible extension work that proposes to study if CryptDB leaks any data patterns.

For the second threat, we start with the assumption that the proxy is not completely secure anymore, and can be subject to an adversary attack that tries to gain control over the entire system. In this second model of CryptDB, we introduce the notion of principals, as types of entities; in our case, the principals are producers and consumers. The types of principals in CryptDB are **external** (i.e., an outside entity that needs first to log in using their application key - usually physical users) and **internal** (i.e., an entity created within the application that doesn't require another key). One example of an external-internal pair of entities is users of an email client and sent emails. In our case, both the mobile-users and the consumers are external principals, requiring an application key (e.g. a password

every the mobile-users needs to renew their anonymous tags, or the aggregator when they need to open a new session for running queries).

In the case of external principals, the proxy server gives each of them a randomly chosen key that represent their access to data. This key is encrypted with the principal's password and stored in a table *external_keys*. Upon principals declaration, one can **delegate access** to another, if specified in the annotated schema.

For instance, in the partial scheme below, there is a physical producer and a physical consumer, as well as their usernames, simplified, *producer* and *consumer*, respectively. The usernames need to inherit access from the physical principals in order to be able to carry out their activities. This delegation is encompassed with the **SPEAKS FOR** annotation. Given two principals, Alice and Bob, if Alice would like to offer Bob access to her data, she will have Bob speak for Alice. This means that Alice's access key is encrypted using Bob's key and stored in the special access table.

PRINCTYPE physical_producer, physical_consumer EXTERNAL;
PRINCTYPE producer, consumer;

CREATE TABLE *producers* (producerid int, username varchar(255),
(username physical user) SPEAKS FOR (userid user));

CREATE TABLE *consumers* (consumerid int, username varchar(255),
(username physical user) SPEAKS FOR (userid user));

Figure 6.1: Annotated schema for creating extenral principals for producers (mobile end-users) and consumers (query aggregators), together with their username tables.

The consumers can be classified into groups, such as commercial, research, local authority, etc., like below. Then, the users can choose what group to join, and the permissions are added to the schema.

CREATE TABLE *consumergroup* (consumerid int, groupid int,
(consumerid user) SPEAKS FOR (groupid group));

Figure 6.2: Grouping consumers into classes, that are later on choose by producers for receiving access for computations over location data.

6.1.2 CryptDB Key Chain Limitations

Unfortunately, CryptDB currently supports access control policies per entity only. There is no policy based on the type of executed query, because of the current lack of encryption protocol for such a requirement. We discuss more on this possible further research point in chapter 7.

6.2 Threat Model

In [SFH10], they define the possible dimensions of an adversary as **means**, **actions**, and **goals**. We deemed this approach to offer a sufficiently complete view of the kinds of attacks a location-aware system can have, and decided to expand on the threat model of $PrivMarket$.

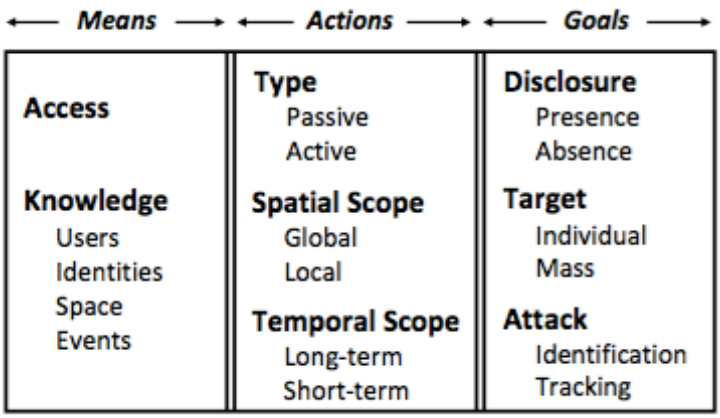

Figure 6.3: The possible dimensions of an adversary. Image courtesy of [SFH10].

Due to limited time constraints on the project, a somewhat shorter threat model is provided. Even though some of the threats might not be currently addressed, while others may have an untested solution, we aimed to provide a view as complete as possible. We proceed with the adversary goal analysis, following the lines of the aforementioned framework from [SFH10], in order to better spot the weak points of $PrivMarket$:

1. **Goals** of an attacker can be partitioned into:

   (a) **Presence vs. absence disclosure** - as reported in 4.5.1, this goal can seriously damage an individual (can lead to theft, individual harassment, and so on). One possible solution might be implementing a **plausible deniability** protocol or adding a **result threshold**. The protocol can probabilistically leave out user data on each collection point, in order to allow a users to deny their presence or absence at a particular spot.
   As for the result threshold, in case it is not achieved, the proxy issues a new query with the requirements loosen (whether it is the area or the time interval). While this can offer protection from an outside threat (after the thresholds are learned, based on the market dynamics), an inside threat (proxy or DBA) might still have free access to the rear-case data. The first solution seems a better approach.

   (b) **Individual vs. mass targets** - an adversary might try to leak information, whether it is an individual basis, or targeted at masses. The plausible deniability can serve as a departure point for investigating on this in more depth.

   (c) **Tracking vs. identification** - An identification attack without side-information should theoretically fail if the system would use a commitment scheme for upload registration, an anonymised network for the data upload per se (such as the Tor network; for more information, please see [MBG+08]). Same for tracking for multiple days. The rest of the cases are subject to further extensions, not currently covered by the project.

2. **Means** of an attack represent the available *technologies* to *Eve*, that can either be **access**, or **knowledge** on the any element or stakeholder of the system.

   (a) The scope of the **access** of an inside or outside eavesdropper is reduced thanks to CryptDB, but it is still a considerable threat on some components:

   - An attack on the DBMS, or a curious DBA cannot get access to the plain texts. Without access to the decryption keys, they cannot even get rid of the more secure onion layers. If, however, a weaker onion layer is the out-most one, such a DET or OPE, the attacker can learn equality or ordinality properties, respectively.

- An attack on the proxy or application layer can seriously endanger the producers and the consumers in the initial solution. With the proposed extension of using key chains, on an entity basis, an attacker will be able to recover the key of the users logged in during the attack, as noticed by [PRZB11] as well. Due to the lack of partitioning on a query basis, this can actually leak a significant user data set, as opposed to a more traditional case supported in [PRZB11] that only leaks a very small set of data.
- An attack on the users' client application (whether it is spoofing it, or taking it over) is to be taken into consideration once an accountability protocol is developed for $PrivMarket$, in order to have an underlying evaluation basis.

(b) For the **knowledge** of an adversary, we covered this under the listing of side-information attacks, in subsection 4.5.1.

3. The **action** analysis of the attacker is partitioned between side-information attacks, and observations that need to rely on a more sophisticated model, with a more dense data set. We currently outlined it here for the completing the criteria analysis.

6.3 Privacy and Anonymity Evaluation

With regard to the previous threat model, the most notable one is an adversary that manages to take control of the proxy and, implicitly, by the logged in users.

Overall, especially on anonymity, the evaluation would have had to entail real data-sets. However, our current artificial data sets lacks authenticity in order to fetch some insightful metrics when trying to quantify anonymity, or even privacy. One option would have been increase the complexity of data generation; however, this would have sliced too generously the remaining time, due to its entailment of further studies of areas not necessarily tangent with the project's.

Chapter 7

Future Work

The research of pragmatic homomorphic systems is still ongoing. The study of privacy and anonymity aware data markets is still an incipient phase. The current project aims to provide valid points for both these subjects. However, the work on each of them, as well as their integration is continuing outside the scope of this project.

For PrivMarket, further analysis needs to be performed in order to provide a holistic view on the capacities and limitations of the system. Initial steps have been made, both design and implementation-wise, but this only leverages a large set of possible extensions and future work. We now present some of them.

7.1 System Future Work

7.1.1 Registration

We briefly described the registration protocol of $PrivMarket$ in subsection 4.5.3. This can be further expanded to a protocol that performs checks upon accepting the data, in order validate the origin of a tuple upload. At that point, the user would have to provide their decommitment value such that the server can check their correctness. However, a more in-depth view is required in order to better understand the feasibility of such a protocol, as well as to derive the best balanced percentage of tuples to be checked, without imposing an overhead on the user's side.

Based on the adopted reward system on which we expand in the upcoming subsections, in might also be the case that the commitment values can provide the basis for a general accountability protocol, on which we discuss next.

7.1.2 Accountability

when dealing with accountability, one needs to verify the quantity and the correctness of the tuples. In the project, we provided accountability based on the semantics of the underlying data sets. However, future work needs to be invested in order to consider a general accountability protocol. An alternative would be the development of a theoretical framework that covers a complete view of the dimensions implied by both the producer and the consumer's accountability, that might include a random spot-check system as introduced in $VPriv$3.1.

A quota system needs to be put in place such that it prevents the users from spamming uploads into the system. One option would be to have the quota defined per journey, and, since the users preserves the anonymous identifier during a single journey, we can propose an extension in which the phone stores all the data points of a journey (for instance, one way travel with the TFL network), and uploads it at once. The server is able to easily verify the journey quota. Then, the server needs to verify the quota

of journeys (that can be for month, for instance). In this case, there can be an intermediate storage layer that gets all journeys that have not been claimed. At the end of the month, the user needs to prove they have not exceeded the journey quota. We assume this can be done via a zero-knowledge proof, similar to the one introduced by PrivStats in 3.3. However, this is solely a departure point for further investigation.

7.1.3 Incentives

The incentive aspect of a data market was not covered by the project's timeline due to time constraints. However, there would be 2 types of payments that can be integrated with PrivMarket. We refer to incentive as financial remuneration; however, the incentive value is driven by the market applicability - for instance, free usage of a software (as we mentioned in subsection, companies can adapt their current online business by offering their services against the participation into the data market), or a voucher system reward. For simplicity, we interchangeably use incentive, reward and remuneration. In this context, there are two types of incentivising systems:

Network Based

One simple payment scheme is a uniform payment promise - all data producers have the probability of earning the same incentive. If all users have probability of 100% of getting a reward, this is a general uniform payment scheme and can be applied without compromising the collected data. All registered accounts are managed separately from the data aggregation protocol. The payment can be relied on any financial currency and be represented by an online payment service provider, whether it is toward a bank account (VISA[1]), or an electronic wallet (PayPal[2]). In the case of the first, we mention that there is a higher maintenance requirement due to the sensitivity of card information, but PrivMarket can simply follow any e-commerce system's design and implementation for this module.

If the probability of a user getting paid is not 100%, we refer to this as being a lottery system. This system has the same requirements as the previous one. The only difference will be that at the end of a predefined period on which all parties agreed, only a subset of users will be paid, based on a random choice. The lottery system does not interfere with our model.

Data-Quality Based

In a data-quality based rewarding scheme, there needs to be a rating system for the producers. It is worth investigating the criteria for rating each mobile user in the market place. However, this will employ social studies, in order to design the system such that it attracts and motivates the mobile users to join the data market. Some basic candidates for this are the frequency of data, or the quality rate of an accountability protocol. They are both depending on the corresponding accountability protocol of the system, and need extensive future work.

Regardless of the payment criteria in data-quality based systems, the payment system can rely on the blockchain technology. Similar to the payment system suggested in [ZNP15], the system integrate a block-chain in order to incentive the users individually, using an e-currency like Bitcoin-www.bitcoin.org/en/. The electronic e-currency wallets are fast to get for each user, and they can be easily changed over time, in order to avoid any meta-information leakage on the user. It is an interesting future work to develop a protocol on this subject, departing from a data-quality based incentive criterion.

[1] www.visa.co.uk
[2] www.paypal.com

Summary of Incentives

There have been some studies on the possible incentives that can motivate the user to willingly sell or share their data, such as [BKSC11, DLA05]. However, they relied on theoretical system. We assume that further studies can be based on a mock market which can be developed starting from our proof of concept. This is certainly a valuable area to be developed.

7.1.4 Add More Aggregates and Extend Tuples

Currently, VPriv supports a basic set of count-based aggregates, but a rapid extention would be to simply allow more aggregates supported by CryptDB as well. If the query is already supported by CryptDB, *PrivMarket* can easily be extended with it, both on the design and the implementation levels. The only further requirement would be to study if the new query provides the needed guarantees in terms of the users' privacy and anonymity.

Tupload tuples can also be extended with a *content* element that provides some user-based input, not necessarily connected to the location per se, but the user. Since this is given by the type of market place, we have not included this under the current scope of the project, and we leave it for further investigation.
To be noted that location data is one of the highest sensitive data that is still not considered personally identifiable information (PII) by the standards. Thus, the user data market can be easily extended to other non-PII information, once location data is solved.

7.2 CryptDB Future Work

There is also a set of possible future work to be done on CryptDB that would bring benefits to our project. We discuss the most relevant below.

7.2.1 Key Chain Management

We've previously seen how CryptDB does the key chain in the case of an untrusted proxy. However, further investigation needs to be done to find other access control strategies. One option might be to allow the proxy to execute an aggregator's query only if the users agreed to whitelist them. Enigma, [ZNP15] relies on secure multi-party computations, and this might a suitable departing point when conduction research on this subject.

An alternative would be to modify the database schema to allow right delegation only over certain queries. The consumers can be clustered on an institution, or intention base, such as commercial, research, governmental or other purposes. Then each mobile users chooses which cluster they are comfortable with to run statistical computations over their data. This would add them to the corresponding cluster upon registration, and delegate the rights to the chosen aggregators. This can still provide the proxy a level of trust too high, but it will smoothly provide an end-to-end working system, excluding the registration and the accountability phase.

7.2.2 Extend Supported Queries

Currently, CryptDB does not offer any computations over negative integers, and no regex (regular expression) searches, but we deem them to be included in the next round of the system's expansion. For the purpose of our system, the supported SQL queries fit our goals. The average statistics are not currently covered by CryptDB, but this is handled at the proxy level by quering a sum and then a count of the involved elements.

However, some customized databases may need more operations supported, depending on the extra content the uploaded tuple can have. For instance, a data market that would like to study the individuals' accessed websites by location, will require the system to get location-based tuples that are issued when the user accesses a website. For optimising the overhead on the users' mobile phone, we can assume their client application collects multiple websites and uploads them on a time-frame basis, by taking their averaged location. In this scenario, a regex search query might be desirable in order to integrate with the current infrastructure.

7.2.3 CryptDB Data Patterns

Since the pragmatic homomorphic encryptions are still in development, to the best of our knowledge, there are no papers that study if CryptDB's onion encryption leaks any data patterns, especially if the RND layer is removed on a column. The result of such an experiment will clearly (in)validate if CryptDB is a sensible choice for our model. However, we consider that at this moment CryptDB's advantages surpasses the disadvantages in order to get a foundation for data markets' privacy and anonymity guarantees.

7.2.4 Map-Reduce with CryptDB

Another interesting extension to make is to design a map-reduce approach for CryptDB in order to study its scalability. A map-reduce paradigm implies paralelizing a large dataset on multiple server instances, as well as the computations to be run. Each server instance runs the query, for instance, a count aggregation, on its piece of data; then a *reduce* function is applied on all collected results, in our case a SUM query on the partial results.

We note that in the context of a simple-case, similar to our approach, each CryptDB instance can have its own proxy and manages their own encryption keys. However, assuming that none of the instance proxies are completely secure, it is worth exploring new mechanisms for the key management.

Chapter 8

Conclusions

We now summarise the work presented in the report.

We departed with the analysis of the considerations and the design of two systems, $VPriv$ and $PrivStats$ that run computations on users' locations. While the locations of the users are generally not revealed, the first is suitable for functions of individual traces, while the second is targeted for statistical functions over collective traces. We weakened the limitation of the second system and formalized a new model, $PrivMarket$, that aims to run statistical computations over the **daily traces** of the users.

We approached the system design by applying two frameworks targeted for location-aware systems, and we define the goal of the system as **achieving location privacy in the presence of side-information** (i.e. system out-of-bound information available to an adversary).

We then provided a case-study, by implementing a potential data market for the users of the TFL network. We built it on an artificial data set that emulates the weekday and weekend journeys 100,000 users over a period of 14 days. The set of supported queries included basic count aggregates. We integrated a user interface for the easiness of querying the data.

Finally, we provided an evaluation of CryptDB, with respect to its design and utility within $PrivMarket$, and we instantiated various departure points for future work.

In conclusion, we contributed to our initial quest of data markets that preserve the producers' privacy and anonimity, by analysing a model that computes aggregated queries on historical traces, and by implementing a proof of concept. The latter helped us to pragmatically get a grasp of CryptDB, a specialised encryption extension for DBMS, that allows running encrypted MySQL queries on encrypted data.

Bibliography

[BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam Oneill. Order-preserving symmetric encryption. In *Advances in Cryptology-EUROCRYPT 2009*, pages 224–241. Springer, 2009. Available from: `http://link.springer.com/chapter/10.1007/978-3-642-01001-9_13` [Accessed 28th May 2016].

[BKSC11] Michael Benisch, Patrick Gage Kelley, Norman Sadeh, and Lorrie Faith Cranor. Capturing location-privacy preferences: quantifying accuracy and user-burden tradeoffs. *Personal and Ubiquitous Computing*, 15(7):679–694, 2011. Available from: `http://link.springer.com/article/10.1007/s00779-010-0346-0` [Accessed 28th May 2016].

[BL90] Josh Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *Proceedings on Advances in cryptology*, pages 27–35. Springer-Verlag New York, Inc., 1990. Available from: `http://dl.acm.org/citation.cfm?id=88328` [Accessed 30th May 2016].

[Bri89] Ernest F Brickell. Some ideal secret sharing schemes. In *Advances in CryptologyEUROCRYPT89*, pages 468–475. Springer, 1989. Available from: `http://link.springer.com/chapter/10.1007/3-540-46885-4_45` [Accessed 20th May 2016].

[cit15] Citizenme. Available from: `www.citizenme.com/`, 2015. [Accessed 24th January 2016].

[Dam14] Maria Luisa Damiani. Location privacy models in mobile applications: conceptual view and research directions. *Geoinformatica*, 18(4):819–842, 2014. Available from: `http://link.springer.com/article/10.1007/s10707-014-0205-7` [Accessed 20th January 2016].

[dat16] Datacoup. Available from: `www.datacoup.com/`, 2016. [Accessed 24th January 2016].

[dig16] Digime. Available from: `https://get.digi.me/`, 2016. [Accessed 24th January 2016].

[DK06] Matt Duckham and Lars Kulik. Location privacy and location-aware computing. *Dynamic & mobile GIS: investigating change in space and time*, 3:35–51, 2006. Available from: `http://www.geosensor.net/papers/duckham06.IGIS.pdf` [Accessed 24th May 2016].

[DLA05] George Danezis, Stephen Lewis, and Ross J Anderson. How much is location privacy worth? In *WEIS*, volume 5. Citeseer, 2005. Available from: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.335.5193&rep=rep1&type=pdf` [Accessed 16th March 2016].

[dMHVB13] Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3, 2013. Available from: `www.nature.com/articles/srep01376` [Accessed 20th May 2016].

[DV11] Subhankar Dhar and Upkar Varshney. Challenges and business models for mobile location-based services and advertising. *Communications of the ACM*, 54(5):121–128, 2011. Available from: `http://dl.acm.org/citation.cfm?id=1941515` [Accessed 26th May 2016].

[Dwo08] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and applications of models of computation*, pages 1–19. Springer, 2008. Available from: http://link.springer.com/chapter/10.1007/978-3-540-79228-4_1 [Accessed 20th May 2016].

[Gol98] Oded Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*, volume 17. Springer Science & Business Media, 1998. Available from: http://www.wisdom.weizmann.ac.il/~oded/PDF/mcppp-v2.pdf [Accessed 28th May 2016].

[GP09] Philippe Golle and Kurt Partridge. On the anonymity of home/work location pairs. In *Pervasive computing*, pages 390–397. Springer, 2009. Available from: http://link.springer.com/chapter/10.1007%2F978-3-642-01516-8_26 [Accessed 30th May 2016].

[Har15] Bill Hardekopf. The big data breaches of 2014. Available from: http://www.forbes.com/sites/moneybuilder/2015/01/13/the-big-data-breaches-of-2014/#71413d3b3a48, 2015. [Accessed 28th May 2016].

[Kru07] John Krumm. Inference attacks on location tracks. In *Pervasive Computing*, pages 127–143. Springer, 2007. Available from: http://link.springer.com/chapter/10.1007%2F978-3-540-72037-9_8 [Accessed 28th May 2016].

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. In *Automata, Languages and Programming*, pages 486–498. Springer, 2008. Available from: http://link.springer.com/chapter/10.1007/978-3-540-70583-3_40 [Accessed 20th May 2016].

[Leo98] Ulf Leonhardt. *Supporting location-awareness in open distributed systems*. PhD thesis, Citeseer, 1998. Available from: https://www.doc.ic.ac.uk/~jnm/ul_thesis.pdf [Accessed 26th May 2016].

[lis] List of data breaches. Available from: https://en.wikipedia.org/wiki/List_of_data_breaches. Accessed 28th May 2016.

[MBG+08] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the tor network. In *Privacy Enhancing Technologies*, pages 63–76. Springer, 2008. Available from: http://link.springer.com/chapter/10.1007/978-3-540-70630-4_5 [Accessed 6th June 2016].

[Mil85] Victor S Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology-CRYPTO85 Proceedings*, pages 417–426. Springer, 1985.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. Available from: https://bitcoin.org/bitcoin.pdf [Accessed 28th May 2016].

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM (JACM)*, 51(2):231–262, 2004. Available from: http://dl.acm.org/citation.cfm?id=972643 [Accessed 20th May 2016].

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptologyEUROCRYPT99*, pages 223–238. Springer, 1999. Available from: http://link.springer.com/chapter/10.1007/3-540-48910-X_16 [Accessed 28th May 2016].

[PBB09] Raluca A Popa, Hari Balakrishnan, and Andrew J Blumberg. Vpriv: Protecting privacy in location-based vehicular services. 2009. Available from: http://static.usenix.org/events/sec09/tech/full_papers/popa.pdf [Accessed 14th March 2016].

[PBBL11] Raluca Ada Popa, Andrew J Blumberg, Hari Balakrishnan, and Frank H Li. Privacy and accountability for location-based aggregate statistics. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 653–666. ACM, 2011. Available from: http://dl.acm.org/citation.cfm?id=2046781 [Accessed 14th March 2016].

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in CryptologyCRYPTO91*, pages 129–140. Springer, 1991. Available from: http://link.springer.com/chapter/10.1007/3-540-46766-1_9 [Accessed 20th May 2016].

[peo16] Peopleio. Available from: https://people.io/, 2016. [Accessed 24th January 2016].

[Pie] David Pierce. Location is your most critical data, and everyones watching. Available from: http://www.wired.com/2015/04/location/. [Accessed 1st June 2016].

[Pol15] Leo Polovet. The value of data. Available from: http://codingvc.com/tag/value-of-data, 2015. Accessed 28th May 2016.

[pop]

[Pop10] Raluca Ada Popa. *Provable and practical location privacy for vehicular and mobile systems*. PhD thesis, Massachusetts Institute of Technology, 2010. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.478.9414&rep=rep1&type=pdf [Accessed 28th January 2016].

[PRZB11] Raluca Ada Popa, Catherine 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*, pages 85–100. ACM, 2011. Available from: http://dl.acm.org/citation.cfm?id=2043566 [Accessed 18th January 2016].

[Rab05] Michael O Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005. Available from: https://eprint.iacr.org/2005/187.pdf [Accessed 2nd June 2016].

[SFH10] Reza Shokri, Julien Freudiger, and Jean-Pierre Hubaux. A unified framework for location privacy. Technical report, 2010. Available from: https://infoscience.epfl.ch/record/148708/ [Accessed 18th January 2016].

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. Available from: http://dl.acm.org/citation.cfm?id=359176 [Accessed 20th May 2016].

[Sma03] Nigel Paul Smart. *Cryptography: an introduction*, volume 5. McGraw-Hill New York, 2003. Available from: http://www.zurich.ibm.com/~cca/sft13/smart08chaps2426.pdf [Accessed 20th January 2016].

[STLBH11] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. Quantifying location privacy. In *Security and privacy (sp), 2011 ieee symposium on*, pages 247–262. IEEE, 2011. Available from: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5958033&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5958033 [Accessed 26th January 2016].

[Swe00] Latanya Sweeney. Simple demographics often identify people uniquely. *Health (San Francisco)*, 671:1–34, 2000. Available from: http://ggs684.pbworks.com/w/file/fetch/102393280/Latanya.pdf [Accessed 28th May 2016].

[Swe02] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002. Available from: http://www.worldscientific.com/doi/abs/10.1142/S0218488502001648 [Accessed 20th January 2016].

[SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE, 2000. Available from: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=848445&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D848445 [Accessed 26th May 2016].

[WRB14] Daniel T Wagner, Andrew Rice, and Alastair R Beresford. Device analyzer: Large-scale mobile data collection. *ACM SIGMETRICS Performance Evaluation Review*, 41(4):53–56, 2014. Available from: http://dl.acm.org/citation.cfm?id=2627553 [Accessed 20th May 2016].

[Yao82] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982. Available from: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4568388&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4568388 [Accessed 20th May 2016].

[ZDG+15] Jinyan Zang, Krysta Dummit, James Graves, Paul Lisker, and Latanya Sweeney. Who knows what about me? a survey of behind the scenes personal data sharing to third parties by mobile apps. Available from: http://jots.pub/a/2015103001/, 2015. Accessed 30th May 2016.

[ZNP15] Guy Zyskind, Oz Nathan, and Alex Pentland. Enigma: Decentralized computation platform with guaranteed privacy. *arXiv preprint arXiv:1506.03471*, 2015. Available from: http://arxiv.org/abs/1506.03471 [Accessed 20th January 2016].