UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

PRIVACIDAD DE UBICACIÓN PARA UN SISTEMA DE MONITOREO DE LA CALIDAD DE ACCESO A INTERNET MÓVIL / *LOCATION PRIVACY FOR A MONITORING SYSTEM OF THE QUALITY OF ACCESS TO MOBILE INTERNET.*

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERÍA CIVIL EN COMPUTACIÓN

GISELLE ALEJANDRA FONT BREVIS

PROFESOR GUÍA:
JAVIER BUSTOS JIMÉNEZ, ALEJANDRO HEVIA ANGULO

MIEMBROS DE LA COMISIÓN:
JOCELYN SIMMONDS WAGEMANN
FELIPE LALANNE ROJAS
NICOLÁS HIDALGO CASTILLO

Este trabajo ha sido parcialmente financiado por NIC Chile Research Labs.

SANTIAGO DE CHILE
2015

Resumen

Con el objeto de medir la calidad de acceso a Internet móvil, NIC Chile *Research Labs* desarrolló *Adkintun Mobile*, un monitor pasivo instalado en los celulares de usuarios voluntarios. Periódicamente, la aplicación registra datos relativos al estado de la red, los que son enviados a un servidor que los recolecta. Los investigadores del laboratorio tienen acceso a los datos almacenados por el colector.

A partir de la conexión a las antenas de celulares, la ubicación del dispositivo puede ser deducida, por lo que la ubicación de los usuarios queda expuesta a la vista de los investigadores, lo que resulta preocupante desde el punto de vista de la privacidad de ubicación de los individuos. Más aún, sólo cuatro puntos espacio temporales son suficientes para re-identificar al 95% de la población a partir de una base de datos anonimizada. Es por ello que este trabajo se enfoca en resolver el problema usando un enfoque criptográfico.

Se propone un modelo en el que los investigadores pueden acceder, consultar y calcular agregaciones sobre los datos almacenados, pero sólo obteniendo de la ubicación de los individuos el resultado de las agregaciones. El modelo utiliza encriptación homomórfica para resguardar la privacidad de ubicación. La información relativa a la ubicación es enviada encriptada desde los celulares hacia el servidor. El servidor puede calcular homomórficamente funciones predefinidas, como contar el número de usuarios en un determinado lugar. Las consultas a la base de datos y la desencriptación, se ejecutan en capas separadas, para evitar que la llave secreta sea utilizada en la desencriptación directa de los datos.

Se implementaron dos versiones de la capa de privacidad de ubicación, con encriptación completamente homomórfica (FHE) usando el esquema BGV, y con encriptación parcialmente homomórfica (PHE) usando el esquema Paillier. El desempeño y *overhead* del sistema, muestran que el modelo es adecuado para cálculo *offline* de estadísticas.

Las contribuciones de este trabajo consisten en proponer una aplicación práctica de FHE para privacidad de ubicación; y discutir sobre el *trade-off* entre privacidad de ubicación y el desempeño del sistema en ambas implementaciones (FHE y PHE).

A todas las mujeres que (anónimamente) han contribuido a forjar la historia de la humanidad.

Abstract

In order to measure the quality of access to mobile Internet, NIC Chile Research Labs developed Adkintun Mobile, a passive monitor installed in volunteer users' mobile phones. Periodically, the client application records data related to network state, which is sent to the collector server. Researchers of the laboratory have access to such stored data.

Since from connexion to antennas location can be deduced, location data of individuals is exposed to researchers, which is a concern for location privacy. Moreover, as only four spatio-temporal points are enough to identify 95% of the population in an anonymized dataset, this work takes a cryptographic approach to solve the problem.

We propose a model where researchers can access, query and compute aggregations on stored data, learning nothing more about users' location than the result of the aggregation. Our model uses homomorphic encryption to preserve location privacy. Location data is sent encrypted from mobile devices to the server. The server can homomorphically evaluate predefined functions such as counting the number of users in a given location. Query and result decryption are performed from a separate layer, which protects the secret key from being used for direct decryption of the records.

We implemented two versions of the location privacy layer using a Leveled Fully Homomorphic encryption (FHE) scheme (BGV), and a Partial (additive) Homomorphic encryption (PHE) scheme (Paillier). The overhead and performance evaluation show that both versions are adequate for offline statistical analysis.

The contribution of this work is to propose a practical use of FHE for location privacy; and to discuss the trade-off between location privacy and system performance for implementations using FHE and PHE.

Acknowledgments

- To NIC Chile Research Labs and the Computer Science Department of University of Chile for the funding of this research.
- To Francisco Cifuentes and Sebastián Blasco for their contributions to this work.
- To all the contributors that daily share their data through the Adkintun Mobile application.
- To my family, for supporting me in all my dreams and new challenges.

Contents

1

Introduction

Where were you on march 19th at 19:51? And on April 22th at 13:11? You probably do not remember. However, do not worry, for sure your mobile phone does. Moreover, if you are lucky enough, your information is stored somewhere on a server that is not really trusted, in a far away country.

Applications that collect users' location are broadly available: from systems interested in aggregated data from a large population (such as traffic monitoring systems) to systems that deliver user centered location-based services (such as searching for nearby "pizza"). Location inference is not only available from devices with a GPS. Location can also be deduced from connection to antennas or wireless networks [20].

Blumberg and Eckersley [4] proposed an enlightening set of sensitive questions that can be answered by querying a locational database:

- *Did you go to an anti-war rally on Tuesday?*
- *A small meeting to plan the rally the week before?*
- *At the house of one "Bob Jackson"?*
- *Did you walk into an abortion clinic?*
- *Did you see an AIDS counselor?*
- *Have you been checking into a motel at lunchtimes?*
- *Why was your secretary with you?*
- *Did you skip lunch to pitch a new invention to a VC? Which one?*
- *Were you the person who anonymously tipped off safety regulators about the rusty machines?*
- *Did you and your VP for sales meet with ACME Ltd on Monday?*
- *Which church do you attend? Which mosque? Which gay bars?*
- *Who is my ex-girlfriend going to dinner with?*

As stated by the authors, the problem is not in the fact that privacy violation was not possible before the rise of ubiquitous and pervasive computing. Private investigators could be hired at hight rates to secretly follow and record the activities of an individual. The difference is that now this is possible at a low cost and for not one, but thousands of mobile device holders at the same time. The ability of applications to track users' location raised a new kind of privacy concern and the development of a field of study which is called *location privacy*.

Different definitions for *location privacy* have been proposed so far:

- the ability to prevent other parties from learning one's current or past location [3].
- a special type of information privacy concerning individuals' claims to determine for themselves when, how, and to what extent location information about them is communicated to others. In short, control of location information is the central issue [13].
- the ability of an individual to move in public space with the expectation that under normal circumstances, their location will not be systematically and secretly recorded for later use [4].

As once location information has been added to a database users lose track of their own data, these definitions suggest that no locations should be collected or stored at all; or at least not in a way that any individuals' location could be deduced at a later date. The sensitive information that could be disclosed from attacks to the database suggests that privacy policy should in this case be ensured not only in terms of a contract, but also through computing mechanisms meant to prevent any leakage.

1.1 Problem definition

This research is about location privacy. In particular, it will focus on the study of the *Adkintun Mobile* application. This section will describe the context of this application and discuss the problem attached to it.

1.1.1 Context

In order to measure the quality of access to Internet from mobile devices, NIC Chile Research Labs developed Adkintun Mobile, a passive monitor installed in volunteer users' smart phones [8]. Data from connections such as bytes sent and received, antenna and timestamps, among others, are gathered by the application. As from antenna connection users' location can be inferred, and as the application is continuously collecting such data, a history of individuals trajectories can also be easily deduced from the database. Protection of users data is then a priority for the research lab: a privacy policy ensures the use of data strictly for research purposes, and that no personal information would be disclosed to third parties.

The system is composed of a set of mobile devices (users) a collector server and a data processor server. Each mobile device has a client application that passively gathers network usage data. Periodically, the client application sends data to the collector server. Once collected, researchers process, analyze data and release research results. Once data is processed, users can retrieve visualizations with statistics about their connections.

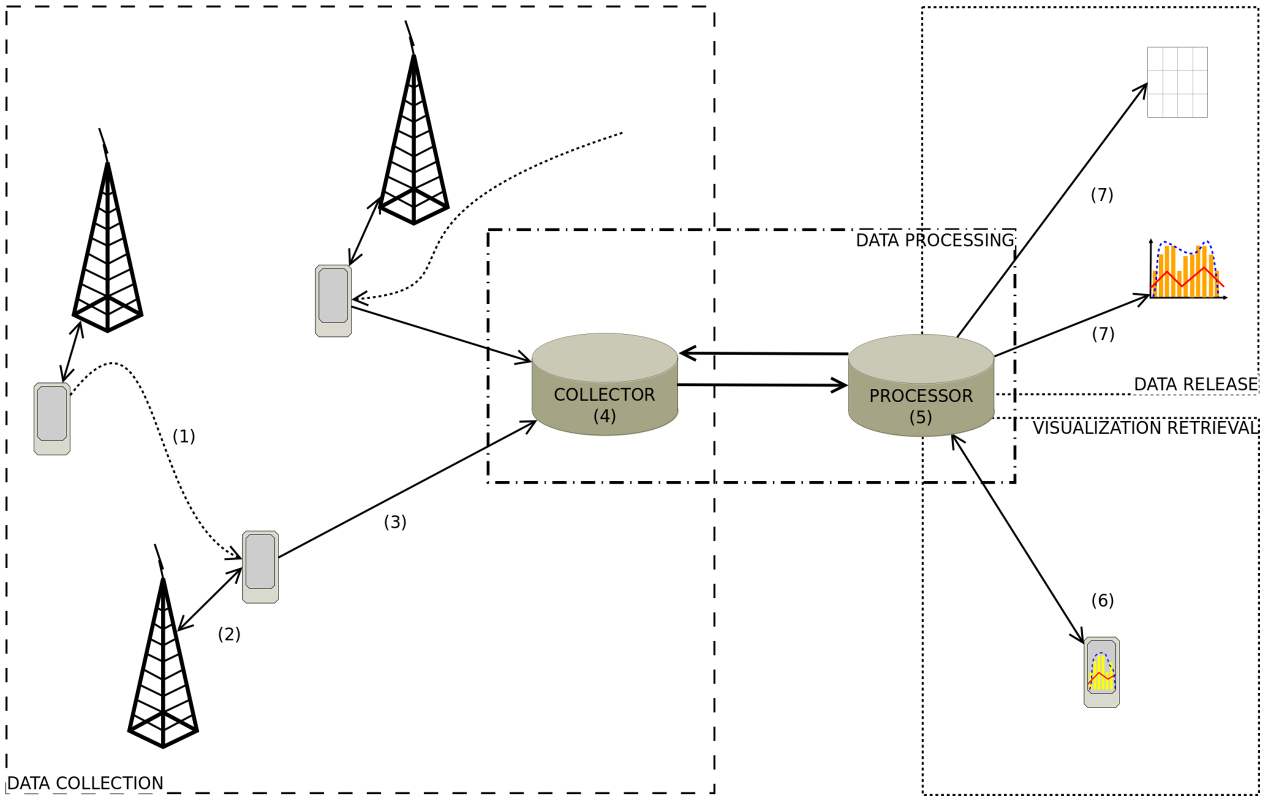

Figure 1.1: Adkintun Mobile system description.

As shown in Figure 1.1, the following is the system description:

Data Collection: Periodically, the client application records data concerning network state, such as bytes sent, received, or signal strength, along with timestamps. While moving (1), mobile devices connect to different antennas (2) in order to get phone and Internet service. Changing from one antenna to another is an event which is also recorded by the application. Other events such as change in the type of service (EDGE, 3G, etc.) are also considered. Periodically, a report is sent (3) to the collector server (4).

Data Processing: In the data processor (5), each record is bound to a specific antenna, using the provided data. Antenna identifiers are used for spatial coordinates and Internet service provider assignation on each record. Data linked in such a way turns to be useful for research purposes.

For data analysis, researchers have full access to data. They have permissions to see any record and query the database.

Researchers are interested in questions that are intrinsically related to location and others that are not. Understanding Internet access behavior in certain areas of the city is related to location; while studying Internet access behavior for different users is not.

Visualization Retrieval: Users can request for visualizations (6) obtained from their own data, such as *How are my bytes sent/received distributed during the day?* or *How is the type of service provided to me distributed during the day?*

Data Release: The system needs to release information, which is basically research results and general data aggregations (7), but never data about a specific user.

1.1.2 The problem

The main problem with this setup is that location data of individuals collected by Adkintun Mobile is exposed to researchers, which is a concern for location privacy.

To be more specific, consider a scenario with *trusted* but *curious* researchers. *Trusted*, meaning that under normal circumstances, they will never publish information belonging to a user. *Curious*, in the sense that they may run algorithms to deduce mobility patterns of specific users. More over, consider that users expect that data collected by the application is about network and Internet access, not about their location or behavior. The following are a couple of issues that rise from this scenario.

On the one hand, Adkintun Mobile systematically stores users' location data. If only owners have access to their data, storing location data would not be a major problem (provided that database is secure under attacks). But *curious* researchers have full access to see and process data, which may lead to users' location privacy violations.

On the other hand, results released as aggregations, do not contain user's identifiers. However, the system may unintentionally leak location information that could be linked to a specific user. This may also violate user's location privacy.

1.1.3 Research question

How can the system described above be refactored, in order to provide location privacy to users, while allowing for broad and efficient data analysis?

More specifically, given a set of functions F that take records as input (each of which bound to a location and a user), and given a user u_{i} whose coordinates are $x_{\mathrm{i}j}$ at time j: Is it possible to hide location information from *curious* researchers, in such a way that they can efficiently compute $f \in F$ but they cannot learn the location $x_{\mathrm{i}j}$ of u_{i}?

2

Related work

This section will discuss related work on location privacy. It will show that privacy concerns that rise from location-aware applications have been largely discussed and solutions have been proposed; however, open problems remain in the field. The section will also discuss how the recent implementation of new cryptographic schemes suggest a different approach to tackle the problem.

Two non cryptographic common approaches to improve location privacy are anonymity and obfuscation. Section 2.1 will review these techniques and discuss their drawbacks. Section 2.2 will introduce the use of homomorphic encryption as a way to preserve location privacy; and review on more detail (*partial* and *fully*) homomorphic encryption properties that allows for calculations over encrypted data.

2.1 Anonymity and obfuscation for location privacy

In a survey of location privacy [20], Krumm reviews anonymity and obfuscation. Anonymity is defined as the ambiguity of identity introduced by the use of pseudonyms or by grouping people together. The intuition is that a service provider may know the location of an entity, but not its real identity. On the other hand, obfuscation is defined as the reduction of data quality. A service provider may or may not know the identity of an entity, but its location remains inaccurate. In the following examples, these tools are used for improving location privacy.

Alastair et al. [3] proposed a framework inspired on anonymous communication techniques for location-based services using Active Badge or Active Bat systems[1]. In this model, the space is partitioned based on the location of the service providers, and the user holds a

[1]Wireless indoor location systems.

Active Badge is a device that periodically transmits a unique infrared signal, which is captured by networked sensors installed in the building. It is meant for locating individuals within buildings.

Active Bat transmits ultrasound signals, which are sensed by a set of receivers mounted on the ceiling of a room. It is meant for determining an individual's 3D position within a room.

set of different pseudonyms to communicate with each of the services. Considering possible collusion between services, the model introduced *mixed zones*, which are zones in between the service areas. In the mixed zones, random pseudonyms are assigned to each user when they access it. This way the services cannot track the path of the user between zones, and so cannot link different pseudonyms to the same user.

In [3], authors also propose methods to quantify the effectiveness of the anonymization and react to possible leaks of privacy. Two cases are considered. The first is when there is a low density of users (or unique user) in contiguous zones: in this case even changing pseudonyms, it is easy to deduce that the user exiting one zone is entering the other. The second is when there are mobility patterns: for example, users that spend more time in some places that identifies them. It is the case of office workers using their desks. Extending this idea to location-based services in urban areas it is possible to find similar problems: in some suburban areas user density may be low; On the other hand, people spend more time in their houses during night, or workplaces during day.

Another example of anonymity usage is presented by Popa et al. [24]. *VPriv* is a location-based vehicular service system for congestion-based road pricing, traffic monitoring, "pay as you go" insurance, among others. The system is composed by one server, many cars equipped with transponders that communicate with the server and a client application that runs cryptographic protocols. While driving, the car records its location and sends a tuple *<tag,time,location>* to the server. The tags are randomly generated by the client application in the registration phase. Drivers are bound to these tags by *commitments*, but this relation is not known by the server. At the end of the month, the client application uses cryptographic protocols with the server, in order to calculate the amount due to the road administration without revealing the tags.

In a traffic context where every vehicle is equipped with the device, the probability of being alone on the road is relatively low. However, in special conditions, such as night driving or driving in low density suburban areas, a user path may be identified even from a set of sequential random-tagged records. As with previously described example, pseudonyms also behave poorly when users are distributed in low density zones, or where mobility patterns are unique for each user.

The idea that anonymized data can be re-identified in some conditions, is captured by the concept of *k-anonymity* [30]. K-anonymity states that if an anonymized database contains rows with a combination of fields that unambiguously characterize an individual, the individual can be re-identified. To preserve privacy, Sweeney proposes that a mechanism should ensure that at least k different rows contain the same combination of pseudo-identifiers. In other words, that each individual should be indistinguishable from at least $k-1$ other individuals.

This concept, combined with obfuscation, or uncertainty in location, are exploited in the *Never Walk Alone* [1] algorithm. They introduce the notion of *(k, δ)-anonymity*, where δ is the imprecision of location due to inaccurate measurements, and the probability of co-location of two distinct moving objects is more probable.

In a similar vein, Bonchi [5] reviews techniques based on k-anonymity and the resulting challenges for publication of locational databases. He discusses how certainty of location and anonymity can be augmented by discarding from the dataset trajectories that violate a predefined *compactness* constraint. Trajectories that cannot be grouped with $k-1$ other trajectories without augmenting the obfuscation of the cluster beyond the compactness value, are considered as outliers and discarded. The compactness parameter gives to the model the certainty of location, while the clustering of trajectories provides the anonymity.

The main problem of applying these concepts to the Adkintun Mobile project is density. Currently there are a few hundred of volunteer users. But even if the number grows to tens of thousands of users, the problem may remain. Considering a large city such as Santiago with 10^5 antennas registered by the system, the probability of being the only user of the system sending information of a particular antenna is not negligible. Periodical patterns repetitions such as going from home to work, can be easily deduced from repeated long stays over night in the same place. In other words, the current user density in Adkintun Mobile does not satisfy the requirements to ensure k-anonymity.

This issue becomes more serious if the traces left by mobility can be likened to individuals' fingerprints. In a study using a dataset where users location was specified hourly by the carrier's antennas, it was shown that 95% of the population could be identified with only four spatio-temporal points [11]. This is the same spacial resolution that Adkintun Mobile has. This leads us to propose that a different approach is needed to ensure location privacy, as next section will show; and avoid solutions that in practice are only superficial changes to the same problem.

2.2 Location privacy and homomorphic encryption

A completely conservative approach to preserve location privacy, is to avoid storing location data. However, this also means that no spatial analysis can be performed. Consider a system such as Adkintun Mobile: no location data means that searching for relationships between location and quality of access to Internet would be beyond the scope of the system. Such a change would ensure location privacy, but at the cost of highly limiting research.

A less conservative approach is to store location data, but encrypted. Encrypted data looks like random data for anyone looking at the database that does not have the secret key. This would hide user's *paths* from *curious* researchers. However, we would like to be able to analyze such data. This is possible for some operations over data, in encryption schemes that have homomorphic properties (as it will be explained later). If operations over location data are limited, such a scheme is achievable.

The idea of storing and operating on encrypted data is used in electronic voting systems [10], secure data aggregation in sensor networks [25] and has been proposed for private data aggregation [28, 18]. Aggregations are computed over ciphers, and the result is later decrypted. Each record remains private and only the returned value is made public. In such a scheme, it is possible, for instance, to count the number of users in a given location without disclosing individuals' location.

In the following, we will explore homomorphic encryption as a tool that allows for both store data encrypted and evaluate a set of functions over it. It was first introduced as *Privacy homomorphism* in 1978 by Rivest et al. [27], after observing that the multiplicative homomorphism property hold on the RSA encryption scheme.

Recalling from introductory algebra, homomorphism is defined as follows:

Definition 2.1 (homomorphism) *Let $\star : A \to A$ and $\odot : B \to B$ be two operations. Let $(A, \star)$, $(B, \odot)$ be two algebraic structures. We say that $f : A \to B$ is an homomorphism if*

$$(\forall x, y \in A) f(x \star y) = f(x) \odot f(y).$$

Applied to cryptography, homomorphic encryption is a scheme in which an operation on plaintext can be mapped to an operation over its ciphertext. In other words, an operation can be computed directly on encrypted data without needing to decrypt it first.

In particular, partial homomorphic encryption (PHE) allows for the evaluation of one type of operation (addition or multiplication, but not both). Note that with additive homomorphism, aggregations can be calculated.

PHE is broadly used in cryptographic protocols using *zero knowledge proofs*[2] [24] and electronic voting systems [2].

Rivest et al. [27] posed the problem of whether it was possible to have a secure homomorphic encryption with not only one, but a set of allowed operations, for example addition and multiplication. As a sample application, the authors proposed a time-sharing data bank service, that would allow storage of encrypted sensitive information, which could be queried without reveling the content to the service provider or foreign attackers.

Since then, many other PHE have been developed. However, the problem posed by Rivest et al. remained open until 2009, when Craig Gentry proposed the first fully homomorphic encryption (FHE) scheme based on ideal lattices [15].

Definition 2.2 *A FHE scheme is such that, given any valid secret key, public key pair*[3] *(s_k, p_k), any circuit*[4] *Ω, plaintext message m_i, and ciphertext $c_i \leftarrow encrypt(p_k, m_i)$, one can compute $c \leftarrow evaluate(p_k, \Omega, c_1, ..., c_t)$, such that $decrypt(s_k, c) = \Omega(m_1, ..., m_t)$.*

The intuition is that in the scheme proposed by Gentry, any function that can be expressed as a circuit Ω, can be evaluated over ciphertext, and the decrypted result would be the same as if the function was evaluated over plaintext[5]. This is possible on a scheme with both

[2] A zero knowledge proof is a cryptographic protocol where two (or more) participants, A and B, interact. A wants to convince B that his data holds some property (proof the property) without disclosing the data to B (B learns nothing about data: zero knowledge is acquired during the proof).

[3] In an asymmetric encryption scheme, a message is encrypted using the public key, and decrypted using the secret key.

[4] A circuit is a model of computation where input values follow a sequence of gates where functions are computed.

[5] To make this notion non-trivial, it also requires that the size of c be "smaller" than the combined size of $\Omega, c_1, ..., c_t$.

additive and multiplicative homomorphism.

The first step proposed by Gentry to make FHE possible was to construct a *somewhat* homomorphic encryption (SWHE) scheme, allowing for evaluation of low degree polynomials homomorphically. This restriction comes from the noise n associated to a ciphertext. The noise corresponds to the ciphertext's random component, and it increases after operations. While adding homomorphically, the noise increases slightly, but while multiplying homomorphically the noise grows much faster. Exceeding some noise threshold N, ciphertext loses the property of being decryptable. The depth of the circuits that can be evaluated with SWHE depends on n (noise) and N (noise threshold) [15].

In the same work, in order to be able to compute an arbitrary number of operations, Gentry proposed bootstrapping as a way of reducing the noise. This was done by adding to the SWHE scheme the ability of homomorphically compute its own decryption function, whose output would be a refreshed[6] version of the ciphertext. Since having this property imposes restrictions to the depth of the decryption circuit and key sizes, another step called squashing would complete the scheme.

Several optimizations and variations of Gentry's original scheme were proposed to obtain a practical and usable FHE. In 2010, Smart and Vercauteren [29] proposed a way to reduce the key and ciphertext size, allowing smaller message expansion during function evaluation. Along with that, they were able to implement SWHE and present timing results. However, due to the unpractical key size associated with the decrypting circuit depth, they were not able to implement the bootstrapping functionality.

After proposing a set of variations to Gentry's scheme, in 2011, Gentry and Halevi presented a working implementation of FHE [16], including the bootstrapping functionality. The implementation was tested on different security levels. Although Lee [21] succeeded on breaking the Gentry-Halevi implementation for the parameters specified in the public challenge posed by IBM Research [26], the lack of efficiency of the FHE implementation was the main drawback of this approach [6]. Another implementation developed by Coron et al. [9] showed similar performance results.

Later, Brakerski and Vaikuntanathan [7] proposed constructing a SWHE scheme based on the Learning With Errors (LWE) problem and introduced a dimension-modulus reduction technique to improve the efficiency of FHE. This work opened the way for implementing the Leveled FHE by Brakerski, Gentry and Vaikuntanathan (BGV), a novel scheme that avoids the use of the bootstrapping step [6].

In 2013, HElib, the first open source library that implements FHE, was released by Halevi and Shoup [19]. It is intended for research on FHE applications. It is based on the BGV scheme [17], and been maintained since then, incorporating Gentry-Halevi-Smart optimization among other variations of the scheme. Written in C++, HElib provides low-level HE routines such as add, multiply and shift.

Since FHE is a field in development, HElib can be seen as a first approach towards the

[6] By refreshed, Gentry means a ciphertext with a level of noise similar to the base level it had when originally encrypted.

practical usage of FHE. Real applications using HElib will find several challenges during implementation, such as data representation, computation overhead and scalability.

3

Proposal

In this chapter, we propose a model for preserving location privacy in Adkintun Mobile, and present the hypothesis and goals of this work.

3.1 Model

In the rest of this work, we adopt the following definition of location privacy:

Definition 3.1 (Location privacy) *Given a dataset X where each record is bound to a location and a user; given a set F of functions $f : X \to \mathbb{R}$ that takes $X^* \subseteq X$ as argument; and curious researchers with access to each record $x_\mathrm{i} \in X$. We say that the system* preserves location privacy *if it reveals nothing more about individuals' location than the result of computing $f(X^*)$, for $f \in F$.*

Intuitively, a system preserves location privacy, if *curious* researchers can learn the result of an aggregation function, but not the location of users used as input in the function evaluation.

Our model uses homomorphic encryption to preserve location privacy as described in Figure 3.1. Antenna identifier (1) is the location data that is encrypted on mobile devices (2), and sent to the collector server along with other plaintext data (3). This means that location data is stored encrypted on the server. The processor server can query the database and retrieve records (4). These records are used as input to homomorphically evaluate a predefined location function (5) that outputs a ciphertext as result (6). The decryptor takes this ciphertext as input, decrypts it (7) and returns the final result of the function evaluation (8).

In this case, the purpose of storing location data is to analyze the relationship between location and quality of access to the mobile Internet. As an example, we would like to take a snapshot of the types of service in an area of the city, and show a spatial representation of these services on a map. In order to do this, we need to count, for each antenna in the area, the number of records that match the different types of service during the interval defined

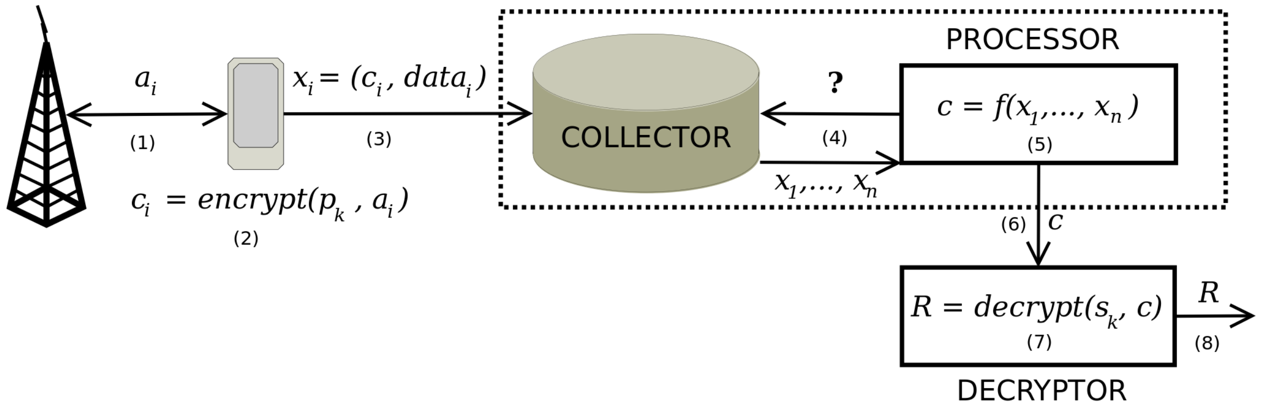

Figure 3.1: Model description.

for the snapshot.

Definition 3.2 (Count) *Given a dataset X with records $x_{\mathrm{i}} = (c_{\mathrm{i}}, \mathrm{data}_{\mathrm{i}})$, $x_{\mathrm{i}} \in X$, where c_{i} is the encrypted location information and* $\mathrm{data}_{\mathrm{i}}$ *are plaintext fields such as type of service, timestamps, or bytes sent/received. Given a subset $X^* \subseteq X$ retrieved from X by a query to X using plaintext fields as filter. Let l_j be a location such as an antenna or city area. We define* count *as the function $count(l_j, X^*)$ that returns the number of records in X^* that correspond to location l_j.*

To evaluate count functions homomorphically, antenna identifiers need to be encoded before being encrypted. The basic idea is that each antenna is represented as a column a_{i}. If the record (row) refers to a connection to antenna a_k then the corresponding value will be 1 (encrypted). Otherwise it will be 0 (encrypted)[1].

The following is an example of its use (note that in the rest of this chapter, we will use plaintext in the examples instead of ciphertext to clarify the approach). Consider 5 antennas, $a_1, ..., a_5$, and 3 records:

Encrypted data (E):

	a_1	a_2	a_3	a_4	a_5
1	0	0	0	1	0
2	0	0	0	1	0
3	0	1	0	0	0

In the example, records 1 and 2 are related to antenna a_4, while record 3 is related to antenna a_2. Then, to count the number of records bound to antenna a_4, we homomorphically evaluate:

$$count(a_4, X) = \mathrm{de}crypt\left(\sum_{k=1}^{3} E_{k,a_4}\right) = 2$$

Since the number of antennas registered by Adkintun Mobile is on the order of 10^5, ap-

[1]Note that since homomorphic encryption schemes are randomized, ciphertext of different zeros look different from each other. Ciphertexts of zeros and ones are thus indistinguishable.

plying this method directly would be impractical: 10^5 fields would be needed just to encode the location of each record. However, antennas can be grouped into sets s_i, where each set contains the same number α of antennas[2], as shown in Figure 3.2.

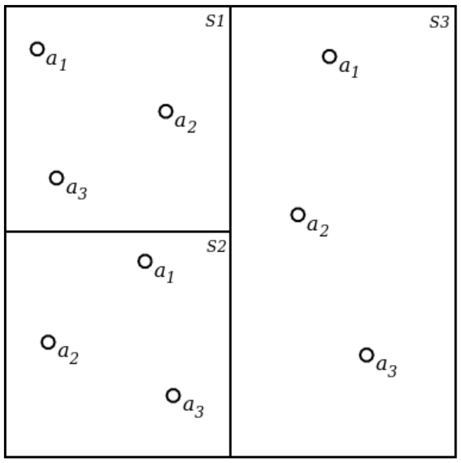

Figure 3.2: Example of sets for $\alpha = 3$.

Definition 3.3 (Public set) *If researchers can learn the set to which a record is bound by inspecting the database, we say that the set is public. In this case the set is stored in plaintext.*

The following is an example of records returned by a query that uses the public set s_1 as filter:

	Encrypted data (E)					Plain-text data			
	a_1	a_2	...	$a_{\alpha-1}$	a_α	set	timestamps	type of service	...
1	0	1	...	0	0	s_1	2013-05-07 10:30:10	3G	...
2	0	1	...	0	0	s_1	2013-05-07 10:30:20	EDGE	...
...	...	...	...	...	...	...	...	...	...
m-1	0	1	...	0	0	s_1	2013-10-07 11:50:30	4G	...
m	0	0	...	1	0	s_1	2013-10-07 11:50:40	3G	...

Public sets can be implemented using both PHE and FHE schemes. Since public sets are stored in plaintext, we can evaluate the count function of a specific antenna $a_i \in s_j$ by filtering for set s_j, as well as other plaintext filters.Then we homomorphically evaluate count over the records obtained by the query:

$$count(a_i, X) = \mathrm{de}crypt\left(\sum_{k=1}^{m} E_{k,a_i}\right)$$

as in the previous example.

[2]Having equally sized sets simplifies the way in which antennas' ciphertexts are handled, as it will be shown later. If α is not a multiple of the total number of antennas, the remaining antennas will be assigned to a set that will be completed with dummies antennas.

Sets could be likened to geographical zones. However, this would disclose location information of a record, since sets can be used as public filters. Another approach is to randomly assign antennas to sets to avoid any relation between sets and locations. Both approaches will be discussed in section 6.1.

Public sets reduce the amount of information stored about antennas, but at the cost of disclosing partial information about users' location. In this case, the set to which the record belongs is disclosed. If this set is large enough, it is reasonable to think that location privacy is not violated. However, using FHE it is also possible to define private zones.

Definition 3.4 (Private zone) *If researchers cannot learn the zone to which a record is bound by inspecting the database, we say that the zone is private. In this case, the zone is encoded and stored encrypted.*

The following is an example that uses both public sets an private zones. Private zones z_k are encoded and encrypted in the same way that antennas: 1 (encrypted) if the record is bound to the zone, or 0 (encrypted) otherwise. The sample records in the example are returned by a query that uses the public set s_3 as filter:

	Encrypted data (E)						Plain-text data			
	a_1	...	a_α	z_1	...	z_α	set	timestamps	type of service	...
1	0	...	0	1	...	0	s_3	2013-05-07 10:30:10	3G	...
2	0	...	0	0	...	1	s_3	2013-05-07 10:30:20	4G	...
...	...	...	...	...	...	...	...	...	...	...
m-1	0	...	1	0	...	0	s_3	2013-05-07 19:40:40	EDGE	...
m	0	...	1	1	...	0	s_3	2013-05-07 19:40:50	3G	...

Private zones can be implemented with FHE schemes, but not with PHE. The reason is that homomorphic addition and multiplication are required in order to compute the count function. Since private zones are stored in ciphertext, in order to evaluate the count function of a specific antenna, e.g. antenna a_i of zone z_ℓ in set s_j, we first filter the database by set s_j, as well as other plaintext filters. Then we homomorphically evaluate the count function over the records obtained by the query:

$$count(a_i, z_\ell, X) = \mathrm{de}crypt\left(\sum_{k=1}^{m} E_{k,a_i} E_{k,z_\ell}\right).$$

Note that since zones are private, a zone can correspond to a geographical zone without disclosing individuals' location information. This approach is useful to count the number of connections in a city area (instead of just connections to a given antenna) as it will be discussed in sections 6.1 and 8.3.

The model can be extended to support many levels λ of private zones, as Figure 3.3 shows.

The number of private zone levels λ defines the number of multiplications required to compute the count function for a given antenna. In this case, we homomorphically evaluate the function:

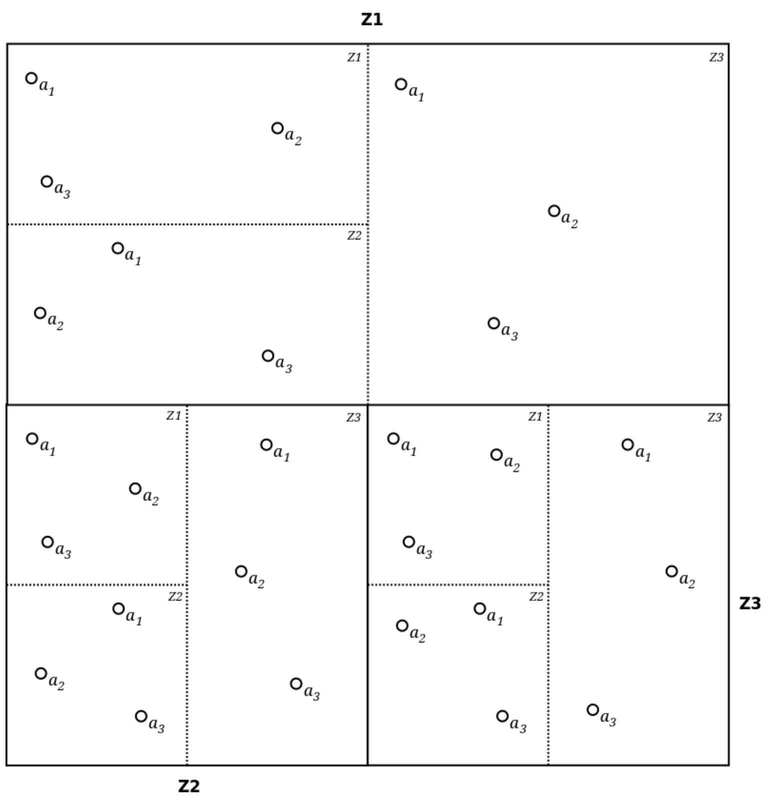

Figure 3.3: Example of zones and subzones, for $\alpha = 3$ and $\lambda = 2$.

$$count(a_{\rm i}, z_J, X) = {\rm de}crypt\left(\sum_{k=1}^{m} E_{k,a_{\rm i}} \prod_{\ell=1, j\in J}^{\lambda} E_{k,z_{\ell,j}}\right),$$

where J is the set of indexes for each zone level.

In summary, the parameters for our model are:

- α: the number of antennas per set (public set or public zone);
- λ: the number of private zone levels (only for FHE);
- π: the number of public sets.

3.2 Hypothesis

Our model provides location privacy while allowing the efficient computation of a predefined set of count functions.

3.3 Goals

Main goal

Determine under which values of parameters α, λ, π the model preserves location privacy while allowing for efficient computations.

Specific goals

- Determine experimentally the values of parameters α, λ, π necessary for preserving location privacy and for efficiently:
  - encrypting data on mobile devices;
  - evaluating count location functions on the processor server;

  for both PHE and FHE schemes;
- Compare the trade-off between location privacy and system performance in both PHE and FHE schemes;
- Quantify the overhead (memory footprint) of both solutions;
- Suggest possible extensions of the model for functions other than counting.

4

Security model

Our solution considers two versions of the system: one using PHE (Additive homomorphism) that we call the *Basic* model; another that uses FHE (additive and multiplicative homomorphism) that we call the *Enhanc*ed model. Our *Basic* model uses the Paillier cryptosystem [23], while our *Enhanc*ed model uses the Brakerski-Gentry-Vaikuntanathan (BGV) cryptosystem [17]. Further details about both cryptosystems will be presented in Section 5.2.

This chapter formalizes the security model and reviews the security assumptions behind our solution.

Our proposed solution has the following general properties:

Dynamic: Our solution allows new users to enter the system, and existing users can leave the system, without compromising the integrity of the data collection or computation processes.

Non-Interactive: Our solution does not require that users interact with each other.

Other properties of the solution are discussed in more detail in the following sections.

4.1 Participants in the protocol

As described in the problem definition, we have two kind of participants in the protocol:

Users: We assume that they are honest, which means that they always submit correct input to the collector. Users can only retrieve their own partial information from the system, or information that is publicly available.

Researchers: We assume that they are curious, which means that they are interested in learning users' location. But they are trusted: they will never publish location information of individuals. Researchers have access to the collector-processor, where location data is stored encrypted and operated homomorphically (to perform the count function); and to the decryptor which takes as input the encrypted result of the count function and a location, and returns the decrypted result for this location. Finally, researchers can also act as users, as they can download the application on their own devices, or simply take control over users' devices (corrupt them).

We assume that the collector-processor can collude with users; that the decryptor can collude with users; Yet the collector-processor cannot collude with the decryptor. We will understand by corrupted user the one that gives away his location information, and by honest user the one that is not corrupted (i.e. wants to keep private his location information) and submits correct information to the collector.

4.2 Privacy definitions

Following the privacy definitions developed in [22], we present two formalizations. Both refer to researchers' active attempts to learn users' locations. The first one refers to privacy against the collector-processor, named collector-processor obliviousness (CPO); the second refers to privacy against the decryptor, called decryptor obliviousness (DO).

Collector-processor obliviousness guarantees that an adversary A_{CPO} cannot infer any information about individuals' locations from corrupting the collector-processor, neither from the messages it receives directly from users, nor from the information stored and processed in the collector-processor, nor from the protocol exchange between the collector-processor and the decryptor. It also ensures that, even if the adversary colludes with an arbitrary number of users, A_{CPO} will not learn any additional information about honest users' locations.

Decryptor obliviousness on the other hand, guarantees that an adversary A_{DO} learns nothing more about users' locations than what can be inferred from the result of the count function. It also ensures that, even if A_{DO} colludes with an arbitrary number of users, he will only be able to learn the result of the count function and anything that can be inferred from that value, but nothing more, from honest users.

4.2.1 Collector-processor obliviousness

To formally capture the the capabilities of an adversary A_{CPO} which corrupts the collector-processor, we consider a model where A_{CPO} has access to the following oracles[1]:

O_{Setup}: When invoked, it returns p_k, which is the public key of the system.

$O_{Encrypt}$: When invoked with an antenna identifier a_i and time t_i, it returns c_i, which is the encryption of the encoding of a_i under public key p_k.

[1] An oracle is a black-box that can be invoked by an adversary. An oracle takes an input (possibly empty), computes some operations and returns an output. The operations performed by the oracle are usually related with an encryption scheme, and common outputs are the encryption of a message, or the public parameters of the cryptosystem.

$O_{Corrupt}$: When invoked with a user u_i at time t_i, it returns the antenna a_i corresponding with the location of u_i at t_i, and any private information associated with u_i.

O_{CPO}: When invoked with two sets of users, their locations (antennas) L_0 and L_1 at time t_i, and an antenna a^*, this oracle flips a coin $b \in \{0,1\}$, computes C_b (which is the set of encryptions c_i for each antenna $a_i \in L_b$ at time t_i), and computes $c_b \leftarrow count(a^*, C_b)$ which is the result of evaluating the count function of antenna a^* in C_b. Finally, it returns (C_b, c_b).

In the *CPO* game, A_{CPO} has access to these oracles in a learning and a challenge phase.

In the learning phase, A_{CPO} can call O_{Setup}, $O_{Encrypt}$, $O_{Corrupt}$ as many times as he wants.

In the challenge phase, A_{CPO} selects two sets of users that where not compromised in the learning phase, and their sets of locations L_0 and L_1 at time t_i such that $|L_0| = |L_1|$. A_{CPO} also specifies an antenna a^*, and then queries O_{CPO} with inputs L_0, L_1, and a^*. O_{CPO} flips a coin $b \in \{0,1\}$ and returns (C_b, c_b) as described above. A_{CPO} outputs a guess b^* of b.

We say that A_{CPO} succeeds in the collector-processor obliviousness game if he guesses b^* such that $b^* = b$.

Definition 4.1 (Collector-processor obliviousness) *A protocol is said to ensure collector-processor obliviousness if for any adversary A_{CPO} that plays the CPO game,*

$$Pr[b = b^*] \leq \frac{1}{2} + \varepsilon \tag{4.1}$$

where ε is a negligible function, called the advantage of A_{CPO}.

This means that essentially A_{CPO} cannot guess b better than flipping himself a coin, which also means that the advantage obtained by A_{CPO} in the game is negligible.

4.2.2 Decryptor obliviousness

To formally capture the the capabilities of an adversary A_{DO} which corrupts the decryptor, we consider a model where A_{DO} has access to the following oracles:

O_{Setup}: When invoked, it returns p_k and s_k, which are the public and secret key of the system.

$O_{Encrypt}$: When invoked with an antenna identifier a_i at time t_i, it returns c_i, which is the encryption of the encoding of a_i under public key p_k.

$O_{Corrupt}$: When invoked with a user u_i at time t_i, it returns the antenna a_i corresponding with the location of u_i at t_i, and any private information associated with u_i.

O_{DO}: When invoked with two sets of users, their locations (antennas) L_0 and L_1 at time t_i, and an antenna a^*, this oracle flips a coin $b \in \{0,1\}$, computes C_b (which is the set of encryptions c_i for each antenna a_i at time t_i from L_b), and computes $c_b \leftarrow count(a^*, C_b)$

which is the result of evaluating the count function of antenna a^* in C_b. Then, it computes $n_b \leftarrow Decrypt(n_b)$, which is the decrypted result of the count function. Finally, O_{DO} returns (c_b, n_b);

In the DO game, A_{DO} has access to these oracles in a learning and a challenge phase.

In the learning phase, A_{DO} can call O_{Setup}, $O_{Encrypt}$, $O_{Corrupt}$ as many times as he wants.

In the challenge phase, A_{DO} selects two sets of users that where not compromised in the learning phase, and their set of locations L_0 and L_1 at time t_i such that for each antenna $a_i : count(a_i, L_0) = count(a_i, L_1)$. A_{DO} specifies an antenna a^* and then queries O_{DO} with inputs L_0, L_1 and a^*. O_{DO} flips a coin $b \in \{0, 1\}$ and returns (c_b, n_b) as described above. A_{DO} outputs a guess b^* of b.

We say that A_{DO} succeeds in the decryptor obliviousness game if he guesses b^* such that $b^* = b$.

Definition 4.2 (Decryptor obliviousness) *A protocol is said to ensure decryptor obliviousness if for any adversary A_{DO} that plays the DO game,*

$$Pr[b = b^*] \leq \frac{1}{2} + \varepsilon \tag{4.2}$$

where ε is a negligible function, called the advantage of A_{DO}.

This means that essentially A_{DO} cannot guess b better than flipping himself a coin, which also means that the advantage obtained by A_{DO} in the game is negligible.

4.3 Privacy analysis

In the following sections we will provide the privacy analysis for both the *Basic* and *Enhanc*ed models.

4.3.1 Analysis for the *Basic* model

Our *Basic* model uses the Paillier cryptosystem [23], which is a PHE scheme that has been proved semantically secure [2].

Analysis for *CPO*

Theorem 4.1 *The proposed Basic model ensures collector-processor obliviousness under the semantic security of the Paillier cryptosystem.*

[2]Semantically secure: the cryptosystem has the property of indistinguishability under chosen plaintext attacks (IND-CPA). See Appendix 10.1.

Proof. Assume that there is an Adversary A_{CPO} that succeeds in the CPO game with a non-negligible advantage ε. We show in what follows that there exists an adversary B that uses A_{CPO} to obtain a non-negligible advantage in breaking the semantic security of the Paillier cryptosystem.

The reduction is direct from the CPO game. B wants to guess if the output from his oracle comes from the Paillier encryption of message m_0 ($b = 0$) or the Paillier encryption of message m_1 ($b = 1$). He uses A_{CPO} as follows:

B simulates O_{CPO} for A_{CPO} using his own oracle. Given sets of locations L_0 and L_1 from A_{CPO}, B computes (n_0, n_1) which are the results of the count function in L_0 and L_1 respectively. Secondly, B computes c_b by querying his own oracle with inputs (n_0, n_1). Thirdly, B computes C_b by querying his own oracle with inputs $(a_{0\text{i}}, a_{1\text{i}})$ for each $a_{0\text{i}} \in L_0$ and each $a_{1\text{i}} \in L_1$. Then B returns (C_b, c_b) to A_{CPO}. After some attempts, A_{CPO} finishes and returns a guess b^*.

B also returns b^* as a guess of b in his own game.

B uses the fact that A_{CPO} has a non-negligible advantage in guessing b to obtain a non-negligible advantage in his own game, and thus breaking the semantic security of the Paillier cryptosystem. As the Paillier cryptosystem has been proven semantically secure, this is a contradiction.

$\therefore$ The collector-processor obliviousness is ensured.

$\square$

Analysis for DO

Theorem 4.2 *The proposed Basic model ensures decryptor obliviousness under the semantic security of the Paillier cryptosystem.*

Proof. As in the proof of theorem 4.1, assume that there is an Adversary A_{DO} that succeeds in the DO game with a non-negligible advantage ε. We show in what follows that there exists an adversary B that uses A_{DO} to obtain a non-negligible advantage in breaking the semantic security of the Paillier cryptosystem.

The reduction is the same as in the CPO proof, but this time B simulates O_{DO} by computing (n_0, n_1) which are the results of the count function in L_0 and L_1, and computes c_b by querying his own oracle with input (n_0, n_1). Then B returns (c_b, n_b) to A_{DO}[3].

The rest of the proof is analogous to the CPO proof. $\square$

[3]Note that B does not need to know b to know the value of n_b, since from construction $n_b = n_0 = n_1$ in the challenge phase.

4.3.2 Analysis for the *Enhanc*ed model

Our *Enhanc*ed model uses the BGV cryptosystem [6], which is a (leveled) FHE scheme that has been proved semantically secure.

Analysis for *CPO*

Theorem 4.3 *The proposed Enhanc*ed *model ensures collector-processor obliviousness under the semantic security of the BGV cryptosystem.*

Proof. Analogous to proof of theorem 4.1, but based on the semantic security of the BGV cryptosystem. □

Analysis for *DO*

Theorem 4.4 *The proposed Enhanc*ed *model ensures decryptor obliviousness under the semantic security of the BGV cryptosystem.*

Proof. Analogous to proof of theorem 4.2, but based on the semantic security of the BGV cryptosystem. □

5

Model implementation

This chapter describes how the model is implemented. The first section presents the modules implemented. We implemented two versions of the model: one using PHE (Additive homomorphism) that we call the *Basic* model; another that uses FHE (additive and multiplicative homomorphism) that we call the *Enhanc*ed model. The following sections review some details of the encryption schemes that were used, to help understand how plaintext and ciphertext are represented and can therefore be homomorphically operated. This will give some intuition about which parameters are important and need to be tuned.

5.1 Module overview

PHE/FHE Android modules: We implemented two modules for mobile devices running on the Android OS. These modules take as input an antenna identifier, that is encoded using a hash map, and returned encrypted under a PHE/FHE scheme.

PHE/FHE server-side processing modules: We implemented two server-side modules. These take as input the result of a query that uses plaintext filters such as time interval, type of service or public zones, and an antenna identifier. These modules evaluate the count function homomorphically, using the hash map associated to the antenna identifier. The result is returned encrypted.

PHE/FHE server-side decrypting modules: We implemented two server-side decrypting modules. These take as input the encrypted result of the count function under the PHE/FHE scheme and return the decrypted value.

PHE/FHE key generation modules: We implemented two modules that generates a public/secret key pair, given the security parameters and others related to the model parameters (α, λ, π).

5.2 Encryption schemes

This section reviews the encryption schemes used and explains details that are relevant for the model implementation. We consider a *Basic* and an *Enhanced* model implementation.

5.2.1 *Basic*

Uses the Paillier cryptosystem, which is a PHE scheme with additive homomorphism [23].

Let $c_1 = encrypt(p_k, m_1)$ and $c_2 = encrypt(p_k, m_2)$ be the encryption of two plaintext messages m_1 and m_2. The homomorphic property that holds on the model is:

$$decrypt(s_k, c_1c_2 \bmod (n^2)) = (m_1 + m_2) mod(n), \tag{5.1}$$

where $n = pq$ and p, q are large secret prime numbers. The property states that a multiplication over ciphertext $(\mathrm{mod}(n^2))$ is equivalent to an addition over plaintext $(\mathrm{mod}(n))$. In this case, plaintext messages are the binary representation of the encoding of an antenna identifier.

We customized an open source library in Java, which primary parameter is b, the bit length of the prime numbers p and q. In the *Basic* implementation only the antenna identifiers are encrypted. Therefore there are no private zones, and antennas are grouped into public sets.

5.2.2 *Enhanced*

Uses the Brakerski-Gentry-Vaikuntanathan (BGV) cryptosystem [17], which is a leveled FHE scheme with additive and multiplicative homomorphism.

Let $c_1 = encrypt(p_k, m_1)$ and $c_2 = encrypt(p_k, m_2)$ be the encryption of two plaintext messages m_1 and m_2.

The homomorphic properties that hold on the model are:

$$decrypt(s_k, c_1c_2) = (m_1m_2) mod(p), \tag{5.2}$$

$$decrypt(s_k, c_1 + c_2) = (m_1 + m_2) mod(p), \tag{5.3}$$

where p is a prime number. The properties state that a multiplication over ciphertext is equivalent to a multiplication over plaintext $(\mathrm{mod}(p))$, and that an addition over ciphertext is equivalent to an addition over plaintext $(\bmod\,(p))$. In this case, plaintext messages are the integer representation of the encoding of an antenna identifier.

We used HElib [19], an open source library in C++, which primary parameters are p, L and k. p is a prime number that represents the modulo of the operations performed over the integers. L is the level that limits the number of operations that can be performed over ciphertext. With $L = 2$ a maximum of one multiplication and a large number of additions can be done. k is the security parameter that represents the number 2^k of attempts required to break the system.

In the *Enhanced* implementation both antenna identifiers and private zones are encrypted. Therefore we have a hierarchical structure where antennas are grouped into private zones and private zones are grouped into public sets.

5.3 Plaintext representation

This section reviews the plaintext representation used by the *Basic* and *Enhanced* models.

5.3.1 *Basic*

Plaintext is represented as an array of bits. Each element of the array is called *block* and has b bits (recall from the last section that b is the bit length parameter of the cryptosystem). Each antenna is represented as a slot which is of size d bits. Then, each block has $\frac{b}{\mathrm{d}}$ slots and can therefore encode $\frac{b}{\mathrm{d}}$ antennas:

$$Slots = \frac{b}{\mathrm{d}} \tag{5.4}$$

Let α be the number of antennas per public set to encode. Then, the number of blocks required to encode the number of antennas per set is

$$Blocks = \frac{\alpha}{\frac{b}{\mathrm{d}}} = \frac{\alpha}{Slots} \tag{5.5}$$

Then to encode the ith antenna we will build a plaintext where the ith slot is a 1, and all the rest are 0s. As each antenna is encoded with d bits, when performing the addition of all the antennas in the count function we avoid overflow by enforcing:

$$count < 2^{\mathrm{d}} \tag{5.6}$$

If $\mathfrak{s}$ is the number of slots per block, a block is then of the form:

$$Block = S_{\mathfrak{s}-1}2^{(\mathfrak{s}-1)\mathrm{d}} + ... + S_2 2^{2\mathrm{d}} + S_1 2^{\mathrm{d}} + S_0 \tag{5.7}$$

where S_{i} is the ith slot of the block.

The idea of such encoding was proposed in a multi-candidate voting system using Paillier [2]. For efficiency, the size of the encoding is set to $O(c|v|)$, with c the number of candidates (equivalent in our case to the number of slots) and $|v|$ the size of the number of voters (equivalent in our case to d).

As an example, let $b = 4$, $\alpha = 6$, $\mathrm{d} = 2$. Then $Slots = \frac{b}{\mathrm{d}} = 2$ and $Blocks = \frac{\alpha}{Slots} = 3$. The encoding of the different antennas will be:

antenna 1:	$[0100, 0000, 0000]$	(first slot of first block)
antenna 2:	$[0001, 0000, 0000]$	(second slot of first block)
antenna 3:	$[0000, 0100, 0000]$	(first slot of second block)
antenna 4:	$[0000, 0001, 0000]$	(second slot of second block)
antenna 5:	$[0000, 0000, 0100]$	(first slot of third block)
antenna 6:	$[0000, 0000, 0001]$	(second slot of third block)

The result of the count function has to be $count < 2^{\mathrm{d}} = 4$ to avoid overflow. So let us say that we add three records associated to antenna 2. The resulting plaintext should be $[0001, 0000, 0000] + [0001, 0000, 0000] + [0001, 0000, 0000] = [0011, 0000, 0000]$. Adding one more will produce overflow to the first slot.

5.3.2 *Enhanc*ed

Plaintext is represented as an array of integers. Each element of the array is called *slot* which maximum value is the integer p (recall from the last section that p is a prime number and also the module parameter of the cryptosystem). Each antenna (or private zone) is represented by a slot. The number of slots in an array depends on a complex relation between p, L, k and other parameters of the cryptosystem, that can be found in the documentation of the library [19].

If α is the number of antennas per zone to encode (or the number of zones per set) we have the relation:

$$\alpha = slots \tag{5.8}$$

In other words α is also the number of slots.

The number of arrays required to encode all the antennas in a public set depends on the number of levels of private zones and therefore on the parameter L of the cryptosystem that limits the maximum number of multiplications that can be performed over the ciphertext.

The model uses one array to encode the antennas, and one array per private zone. For example if L=2 the cryptosystem can perform one multiplication, so only one level of private zones is allowed.

Then, in order to encode the ith antenna of the j^{th} private zone, we built a plaintext of two arrays where the ith slot of the antenna array is 1, and all the rest are 0s, and the j^{th} slot of the private zone array is 1, and all the rest are 0s. As each antenna is encoded with an integer modulo p, when performing the addition of all the antennas in the count function we need to avoid losing the information by enforcing that:

$$count < p \tag{5.9}$$

As an example, let $\alpha = 3$, $L = 2$ and p=5. The encoding of the different antennas will be:

antenna 1 of zone 1:	$[1,0,0]$ $[1,0,0]$
antenna 2 of zone 1:	$[0,1,0]$ $[1,0,0]$
antenna 3 of zone 1:	$[0,0,1]$ $[1,0,0]$
...	...
antenna 1 of zone 3:	$[1,0,0]$ $[0,0,1]$
antenna 2 of zone 3:	$[0,1,0]$ $[0,0,1]$
antenna 3 of zone 3:	$[0,0,1]$ $[0,0,1]$

The result of the count function has to be $count < p = 5$. So let us say that we add four records associated to antenna 2. The resulting plaintext should be $[0,4,0]$. Adding one more will produce $[0,0,0]$, losing information about the antenna.

5.4 Count function

This section reviews how the count function is implemented based on the ciphertext operations available in both the *Basic* and *Enhanc*ed model. In the previous section, we reviewed how antenna and private zone information was encoded on plaintext. In this section, we will explain how the ciphertext is represented and what methods can be used to access its components.

5.4.1 *Basic*

The ciphertext follows a structure similar to the plaintext structure. However, instead of an array of bits, we have an array of ciphertexts where each element of the array corresponds to the same block of the ciphertext. Note though that there is no relation between a plaintext slot and a ciphertext block. As an example, if we had the following plaintext representation of antenna 3:

$[0000, 0100, 0000]$,

an example of how the ciphertext would look like is[1]:

$[6328, 7273, 8449]$.

As it is not possible to access each slot in the ciphertext, if we were interested in counting the number of records associated to the antenna in the ith slot of the j^{th} block, we can instead retrieve the j^{th} block for each record, perform the addition of all these blocks. To retrieve the final result, the resulting block is decrypted and then the ith slot can be extracted and returned.

Following the previous example, if we had three ciphertexts:

[1]This is just an example, not a real ciphertext.

$$[0000, 0100, 0000] : [6328, 7273, 8449]$$
$$[0000, 0001, 0000] : [3412, 1235, 7643]$$
$$[0000, 0100, 0000] : [1263, 1294, 3457]$$

and we want to count the number of records that correspond to antenna 3, which is the 1st slot of the 2nd block, we should:

- Retrieve the 2nd block of each record:
  7273
  1235
  1294
- Perform the addition (that following the homomorphic properties described in the beginning of the chapter, corresponds to a multiplication on the ciphertext):
  7273 x 1235 x 1294 = 11622908570
- Decrypt the result using the secret key:
  1001
- Retrieve the 1st slot:
  10

As this is the binary representation, the result is then 2.

5.4.2 *Enhanc*ed

In this case, the ciphertext does not present similarities in the structure with the plaintext structure. The original array is mapped to a multidimensional array. The library does not provide methods to access the equivalent of the plaintext slot directly in the ciphertext. Both the multiplication and addition over ciphertext are defined over the whole array, where the operation is performed member by member.

In the following example, we will show plaintext instead of ciphertext to clarify the approach. If we had two ciphertext corresponding to two arrays that we wish to operate over, $[1, 0, 0]$ and $[0, 0, 1]$:

- the result of the multiplication would be $[0, 0, 0]$;
- the result of the addition would be $[1, 0, 1]$.

If we want to count the number of records associated to the ith antenna of the j^{th} private zone we need to multiply the ith slot of the antenna array by the j^{th} slot of the zone array. It is not possible to access directly the slot, but the library provides a rotation method, that allows to rotate the array from the j^{th} position to the ith position, so both elements are aligned before the multiplication.

For each record, we first rotate the zone array. Second, we perform the multiplication. Third, all the results are added. The final result is decrypted and the slot we are interested in can be retrieved.

As an example, lets say we have 5 records where we would like to count the number of records corresponding to antenna 2 of zone 3. We need to rotate the zone array in one position to the left. As the ciphertext is a multidimensional structure, we show the equivalence of the operation on plaintext for simplicity. The operation is performed as follows:

$$\begin{array}{lll r}
a:[1,0,0]z:[1,0,0] & rotate(z) \rightarrow a:[1,0,0]z:[0,0,1] & multiply(az) \rightarrow & [0,0,0] \\
a:[0,1,0]z:[0,0,1] & rotate(z) \rightarrow a:[0,1,0]z:[0,1,0] & multiply(az) \rightarrow & [0,1,0] \\
a:[1,0,0]z:[0,0,1] & rotate(z) \rightarrow a:[1,0,0]z:[0,1,0] & multiply(az) \rightarrow & [0,0,0] \\
a:[0,1,0]z:[0,0,1] & rotate(z) \rightarrow a:[0,1,0]z:[0,1,0] & multiply(az) \rightarrow & [0,1,0] \\
a:[1,0,0]z:[0,1,0] & rotate(z) \rightarrow a:[1,0,0]z:[1,0,0] & multiply(az) \rightarrow & [1,0,0] \\
 & & add(results) = & [1,2,0]
\end{array}$$

The result is decrypted then the 2^{nd} slot is retrieved since we are interested in antenna 2. The result of the count function in this example is 2.

6

Parameter selection

The purpose of this chapter is to find practical values for the parameters α, λ and π of our model. These parameters depend on the number of antennas that the system needs to handle, and on the parameters of the *Basic* and *Enhanc*ed models, which provides different levels of performance depending on the selection. Another topic that will also be considered in this chapter is the equivalence in terms of security and privacy of both models, to make both models comparable.

6.1 Antennas

This section reviews details about the number of antennas in the current system, how they are handled, and their geographical distribution. This section will show the relation between α, λ and π, and establish how the antennas should be partitioned.

Antennas are named in the database using three identifiers: *mnc* (service provider), *lac* (local area code) and *c*id (cell identifier). Given these three identifiers, the coordinates of the antenna can be retrieved.

mnc refers to the operator or service provider. It is thus, a public identifier for the purpose of this work, since it does not provide any location information. In our model, *mnc* is stored in plaintext. Table 6.1 shows the number of antennas that each operator manages in the major metropolitan areas of central Chile. This is the more populated zone in Chile, and the more dense. In the table we see that the maximum number of antennas per *mnc* is 20948, for $mnc = 2$. Our model needs to handle at least this number of antennas.

For the *Basic* model, α is the number of antennas per public set, and π the number of public sets. Then the number of antennas per mnc that the model can encode is $\alpha\pi$. The relation between the parameters should then hold:

$$\alpha\pi \geq 20948. \tag{6.1}$$

mnc	n^o of antennas
0	17
1	10903
2	**20948**
3	14387
9	173
10	11708
Total	58136

Table 6.1: Number of antennas in the major metropolitan areas of central Chile by operator.

For the *Enhanc*ed model, α is the number of antennas per private zone (and the number of private zones per public set), λ the number of levels of private zones, and π the number of public sets. Then the number of antennas per mnc that the model can encode is $\alpha^{\lambda+1}\pi$. The relation between the parameters should then hold:

$$\alpha^{\lambda+1}\pi \geq 20948. \tag{6.2}$$

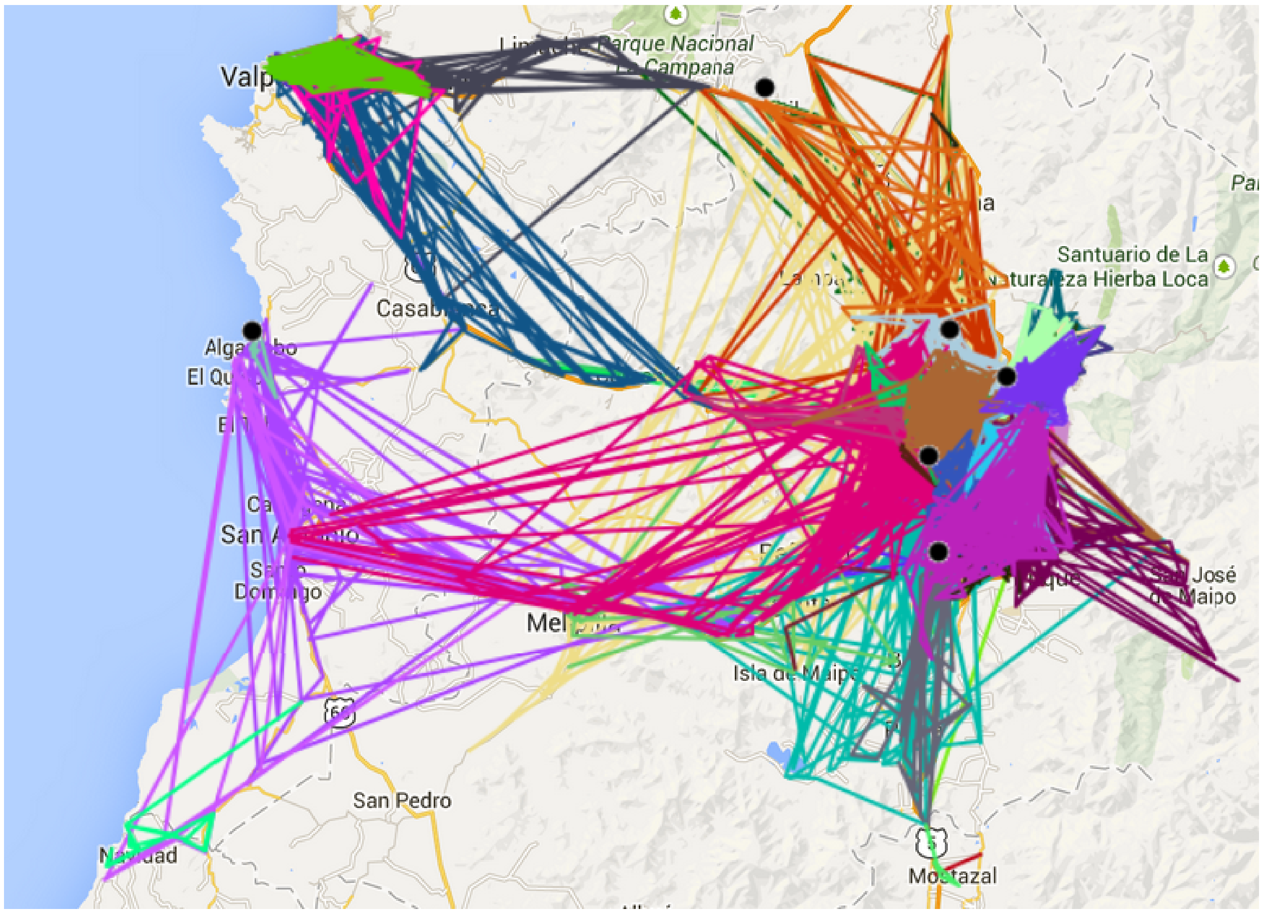

Figure 6.1: Distribution of antennas of operator $mnc = 2$ in the major metropolitan areas of central Chile by local area code.

lac stands for local area code and *c*id for cell identifier. Figure 6.1 shows the distribution of the antennas for $mnc = 2$. Each color represents a different *lac*. The figure shows that even though these zones overlap in some places, antennas grouped by *lac* are geographically grouped. *lac* should then be managed as a private identifier, because it discloses location information about the geographical zone in which the antenna is.

For the assignation of each antenna to a public set, instead of assigning it to the natural set which is *lac*, that would disclose location information of the user, we choose to make this assignation randomly. As shown in Figure 6.2, a random distribution of the antennas into sets (right), discloses less location information than antennas grouped by geographical zone (left). The area enclosed by the set augments, and so does the uncertainty about the location of the individual connected to the antenna.

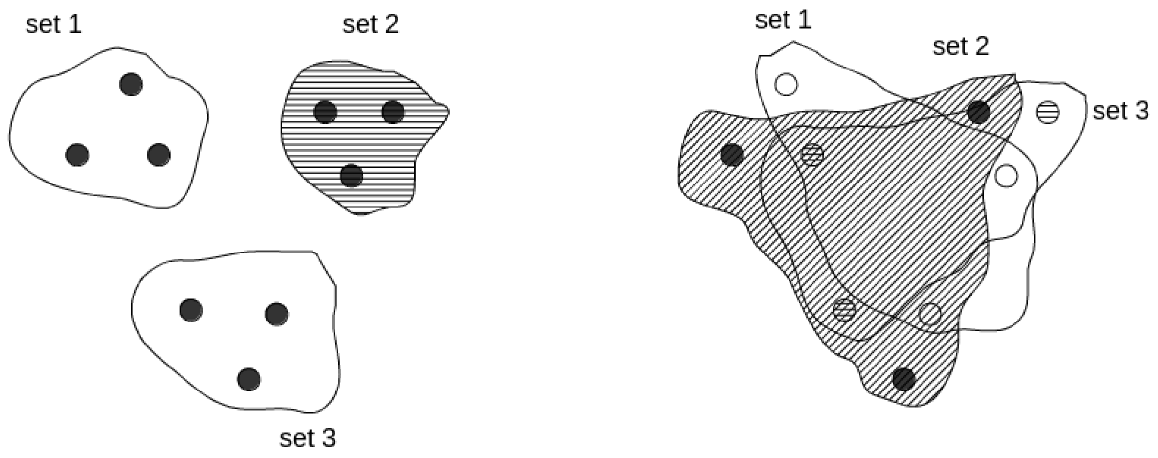

Figure 6.2: Distribution of antennas to sets by geographical zone (left) vs. randomly (right).

As a summary, the assignation of the antennas into π public sets, will be done randomly. The assignation of antennas to private zones can be done both by geographical zone or randomly, as private zones will be encrypted in the database. The resulting trade-off between location privacy and performance of the system, will be discussed in the Discussion chapter (chapter 8).

6.2 Security equivalence

For both *Basic* and *Enhanced* models to be comparable in terms of performance, we need to set equivalent parameters for security. However, security in the Paillier cryptosystem used in the *Basic* model, is expressed in terms of bit length, which is related to the size of the keys. But in the HElib library, used by the *Enhanced* model, security is expressed in terms of k, which refers to the number of attempts (2^k) required to break the system.

To make both security levels equivalent, we followed the recommendations of the ENISA report [14]. This first establishes the equivalence between the two ways of expressing the security level as shown in table 6.2. And second, sets the minimum recommended value to

	b	k
legacy	1024	80
near term 1	2048	112
near term 2	3072	128
long term	15360	256

Table 6.2: ENISA equivalence between security parameters.

100-bit security level[1] (k=100).

Therefore we chose to set our security levels to b=2048 (*Basic* model) and k=112 (*Enhanc*ed model).

The importance of setting the security parameter is also related to the α, λ and π parameters. In particular, for the *Enhanc*ed model, a change in k produces a change in the number of slots handled by the plain-text array, and thus in the number of antennas that can be represented. A change in k also produces a change in the key size and cipher. Fixing k makes it easier to find efficient combinations of the other parameters.

6.3 Location privacy equivalence

This section reviews the relation between the parameters of the *Basic* and *Enhanc*ed models in order to ensure an equivalent level of location privacy.

6.3.1 Relation between parameters

While antennas are stored encrypted in both models, public sets are not. The definition of the size of the public set impacts then in the level of privacy of the model. Regarding the assignation of antennas to public sets, it is direct to establish that both models should have the same assignation: the number of public sets should be the same and the total number of antennas in each public set should be the same. As in the *Enhanc*ed model α is the number of antennas per private zone, and λ the number of levels of private zones, the relation between the parameters of both models is given by:

$$\alpha_{basic} = \alpha_{enhanced}^{\lambda+1}$$
$$\pi_{basic} = \pi_{enhanced}$$

Given a record that belongs to a known public set s, privacy can then be measured as the probability that one encrypted antenna $c_i \leftarrow encrypt(a_i)$, $a_i \in s$ corresponds to a particular antenna $a^* \in s$, which can be expressed as:

[1] Brakerski et al. [17] also recommend a minimum security level of k=100 for the BGV cryptosystem used in our *Enhanc*ed model.

$$Pr[\mathrm{de}crypt(c_\mathrm{i}) = a^*]_s = \frac{1}{|s|} = \frac{1}{\alpha_{basic}} = \frac{1}{\alpha_{\mathrm{e}nhanc\mathrm{ed}}^{\lambda+1}} \tag{6.3}$$

Another parameter that impacts location privacy is related to the maximum number of additions that can be performed in the system without producing overflow (in the *Bas*ic model) or surpassing the module (in the *Enhanc*ed model). This number sets the maximum size of the input (number of records in the query to the database) that can be managed by the count function. For the equivalence between the two models, the relation:

$$2^\mathrm{d} \approx p \tag{6.4}$$

should hold, as 2^d and p express this maximum size of the input for each model.

Then, given a set of records X where we would like to count the number of records that correspond to antenna a^*, privacy can then be expressed as the probability that a particular encrypted antenna $c_\mathrm{i} \leftarrow \mathrm{e}ncrypt(a_\mathrm{i})$, $a_\mathrm{i} \in X$ corresponds to antenna a^*, after knowing that the result of the count function is n_a :

$$Pr[\mathrm{de}crypt(c_\mathrm{i}) = a^*]_{X,n_a} = \frac{n_a}{|X|} \approx \frac{n_a}{2^\mathrm{d}} \approx \frac{n_a}{p} \tag{6.5}$$

In chapter 8, we will discuss how larger sets of records can be handled without the need of augmenting d or p, and reducing the previously defined probability.

Finally, as both events disclose some location privacy information, location privacy can be measured as the sum of the probabilities represented in equations 6.3 and 6.5:

$$Pr[\mathrm{de}crypt(c_\mathrm{i}) = a^*] = Pr[\mathrm{de}crypt(c_\mathrm{i}) = a^*]_s + Pr[\mathrm{de}crypt(c_\mathrm{i}) = a^*]_{X,n_a} = \frac{1}{|s|} + \frac{n_a}{|R|} \tag{6.6}$$

which for the *Bas*ic model can be expressed as:

$$Pr[\mathrm{de}crypt(c_\mathrm{i}) = a^*] = \frac{1}{\alpha} + \frac{n_a}{2^\mathrm{d}}, \tag{6.7}$$

and for the *Enhanc*ed model can be expressed as:

$$Pr[\mathrm{de}crypt(c_\mathrm{i}) = a^*] = \frac{1}{\alpha^{\lambda+1}} + \frac{n_a}{p}. \tag{6.8}$$

6.3.2 Parameter values

For the assignation of values to the model parameters, we will review the *Bas*ic model as it is more simple and the equivalence to the *Enhanc*ed model has already been established.

π	minimum value of α
1	20948
2	10474
3	6983
10	2095
20	1048
100	210

Table 6.3: Minimum value of α given different values of π.

The location privacy of the *Basic* model is expressed by equation 6.7 as:

$$Pr[decrypt(c_{\mathrm{i}}) = a^*] = \frac{1}{\alpha} + \frac{n_a}{2^{\mathrm{d}}}$$

From equation 6.1 we know that

$$\alpha\pi \geq 20948.$$

This means that for different selections of the number of public sets π, the minimum values for α are listed in Table 6.3.

Supposing that $n_a \ll 2^{\mathrm{d}}$, it is enough to take $2^{\mathrm{d}} > \alpha$ in order to avoid increasing significantly the probability due to the second term of the equation.

As α is restricted by the number of slots and blocks of the cipher, it is reasonable to think that more than three public sets will be required to partition the antennas. We can choose then $2^{\mathrm{d}} > 6983$. Taking values of 2^{d} many orders of magnitude bigger than this, will cause an impact in the performance since more blocks will be needed to encode the antennas. Since $2^{12} = 4096$, $2^{13} = 8192$ and $2^{14} = 16384$, it is reasonable to fix $\mathrm{d} = 13$.

The equivalence for the *Enhanced* model would be $p \approx 8192$ from equation 6.4.

6.4 Tuning of α, λ, π

In order to determine values of α, λ, π for which the system allows for efficient computation and preserves location privacy, we checked the parameters of the *Enhanced* model, which are less flexible than the parameters of the *Basic* model.

In the *Enhanced* model, the parameter L defines the number of levels of operations allowed by the system. It is thus related to λ by $L = \lambda + 1$. For a fixed p and k, fixing L also determines the number of slots available in the plaintext array, and thus the number of antennas α that can be encoded per private zone.

For different values of $p \approx 8192$ and L, we measured the number of generated slots, the size of the key and the size of the cipher. For each value of L, p was variated, and the ranges of sizes obtained are shown in Table 6.4.

L	slots min	slots max	key min [MB]	key max [MB]	cipher min [kB]	cipher max [kB]
2	36	3538	2.6	33.5	170	197
3	48	1620	11.1	65.8	373	400
4	59	658	32.4	123.6	651	714

Table 6.4: Range of number of slots, key sizes and cipher sizes generated from variations of p, for different values of L.

The results show that the range of key size and cipher sizes increases with L. A lower level of L will then give a lower overhead of the system and better performance.

Regarding the number of slots and how many antennas can be encoded per public set, note that if we chose from the table $L = 2$ and $slots = 36 = \alpha$, with $\pi = 17$ we would have $17 \text{x} 36^2 = 22032$ antennas per set (> 20948); if we chose $L = 2$ and $slots = 145 = \alpha$, with $\pi = 1$ we would have $145^2 = 21095$ antennas per set (> 20948); and if we chose $L = 3$ and $slots = 48 = \alpha$, with $\pi = 1$ we would have $48^3 = 110592$ antennas per set (> 20948). This means that $L = 2$ or $L = 3$ are enough to encode all the antennas in a public set for $1 < \pi < 17$.

We also run a script to determine if for the given numbers of p, L and k, it was possible to make one rotation, $L - 1$ multiplications and p additions (simulating a count function evaluation) and retrieving the correct decryption of the result. For many combinations, the result of the operations was too noisy and only garbage could be retrieved. For the set of combinations that where correctly decrypted, we chose the one closer to $p \approx 8192$ and with lower key and cipher size.

The choice was $p = 8179$, $L = 2$, which gives $slots = 41$. This means that $\alpha = 41$, $\lambda = 1$ and $\pi = 13$.

The equivalence for the *Basic* model is $\alpha = 41^2 = 1681$ and π=13.

The present tuning is valid for the central area of Chile. For other countries or regions, a different tunning may be suitable. However, the parameter selection allows changes such as the addition of new antennas to the system, without need to change the settings. New public sets can be created to handle the new antennas.

7

Experiments

This chapter describes the experiments and results. Our goal was to evaluate the behavior of the system. The set of experiments is aligned with the goals of this project related to the evaluation of the overhead of our implementation and performance of the system for both versions of the model: $Basic$ and $Enhanc$ed.

7.1 Experimental setup

This section describes the experiments run in both client-side (mobile devices) and server-side modules (collector-processor). The set of experiments is run for the selected parameters that give equivalence between the two models, and that we want to profile and measure their runtime:

$Basic$:

$b = 2048$ bits (bit length), $blocks = 11$, $slots = 157$, d $= 13$ bits (bits per slot).

$Enhanc$ed:

$p = 8179$, $L = 2$, $k = 112$.

For all the experiments, we run experiments on both models and compared the results.

7.1.1 Experiment 1: overhead of the solution

We measured the overhead (memory footprint of the models) in terms of the public key size, secret key size, and ciphertext size. As the antenna identifier is currently stored in the database with two integer identifiers (lac: local area code, cid: cell identifier) which sizes are $32bits + 32bits = 64bits$, any additional storage required by the system can be considered as overhead.

7.1.2 Experiment 2: antenna encryption on mobile phones

We encrypted 100 different antennas chosen at random and measured the execution time in [ms] for each encryption. The experiment was run on two Android smart-phones.

- Motorola XT320 (600 MHz CPU, 512 MB RAM), OS version Gingerbread 2.3.6

- Motorola MotoX8 (1.7GHz Dual-Core Krait CPU, Quad-Core Adreno 320 GPU, 2GB RAM), OS version Kitkat 4.4.4.

7.1.3 Experiment 3: count function evaluation without query size restriction

We evaluated the count function using 100 different antennas chosen at random and measured the execution time in [ms] for each evaluation. The queries where performed on a mongoDB database with real data collected over a month, customized with the antenna identifiers and private zones (when required) encrypted.

The experiment was run on a laptop with the following characteristics:

Lenovo Thinkpad Edge (Intel® Core^{TM} i3-3120M CPU @ 2.50GHz x 4, 3.4GB RAM) with OS Ubuntu 14.04, 64 bits.

7.1.4 Experiment 4: count function evaluation for different query sizes

We evaluated the count function using 100 different antennas chosen at random and measured the execution time in [ms] for each evaluation. The queries where performed on a mongoDB database with real data collected over a month, customized with the antenna identifiers and private zones (when required) encrypted. The experiment was repeated with query results of size 100, 200, ..., 1000, to evaluate the behavior of the system for an increasing number of operations required to evaluate the count function.

The experiment was run on a laptop with the following characteristics:

Lenovo Thinkpad Edge (Intel® Core^{TM} i3-3120M CPU @ 2.50GHz x 4, 3.4GB RAM) with OS Ubuntu 14.04, 64 bits.

7.1.5 Experiment 5: decryption of the final result

We decrypted the result of 100 different count functions, for antennas chosen at random and measured the execution time in [ms] for each decryption.

The experiment was run on a laptop with the following characteristics:

Lenovo Thinkpad Edge (Intel® Core^{TM} i3-3120M CPU @ 2.50GHz x 4, 3.4GB RAM) with OS Ubuntu 14.04, 64 bits.

7.2 Results

The results are presented graphically, showing the comparison between the *Basic* and the *Enhanc*ed model.

Figure 7.1 shows the result of experiment 1, which presents the overhead, measured in size [kB] required to manage the public key, secret key, and cipher for both models. Public and secret key are used during runtime to encrypt and decrypt data. Thus, both should be managed in RAM. Ciphers are also managed in RAM during encryption, but are then stored inside the devices and later in the collector.

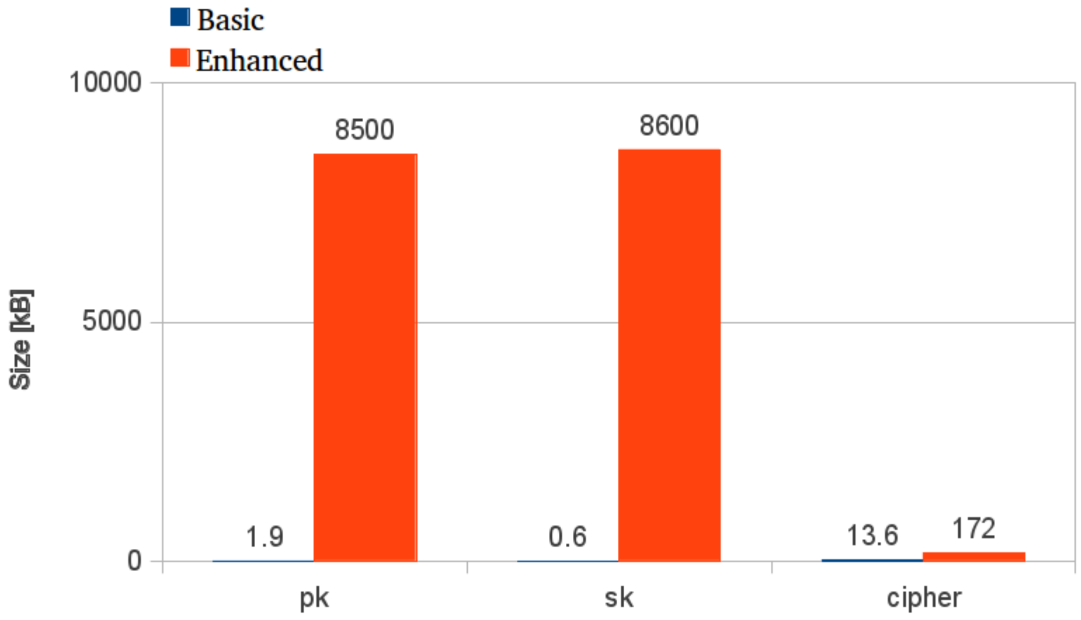

Figure 7.1: Experiment 1 - Overhead (memory footprint) of the solution. The current implementation of Adkintun Mobile (baseline) does not use public or secret key (0kB) and the storage of an antenna identifier is 0.008kB (which can be compared to the cipher size on both models).

The results show that the *Enhanc*ed model requires around 4500 times more memory than the *Basic* model to manage the public key, around 14300 times more for the secret key, and 13 times more for the cipher. However, even if the difference between the two models is significant, keys and ciphers have a size that can be managed by the different parts of the system.

Regarding the client-side module, a public key of 8.5MB represents between 0.5% and 1.6% of the available RAM, which is reasonable. The cipher size corresponds to the size of

model	XT320 [ms]	MotoX8 [ms]
Basic	26835	3788
Enhanced	890	241

Table 7.1: Experiment 2 - Average execution time of the antenna encryption on mobile phones.

the encrypted antennas stored in the device, while they await transmission to the collector. We expect that this activity will occur at least once a week. According to the data stored in our database, each mobile device makes on average 200 antenna switches weekly, which sums up to 200 x 170kB = 34MB of storage in each mobile phone per week. This represents only 1.7% of the storage available on current smart-phones, that usually have more than 2GB internal storage. The transmission of data through WIFI would take a few minutes.

As the secret key is managed on the server-side, a 8.6MB key size cannot be seen as a drawback. However, to store the encrypted data, additional resources have to be considered. In the current implementation of Adkintun Mobile, each antenna identifier uses 64 bits (8 bytes). The antenna identifiers represent 0.4% of the storage used by the whole database. Compared to this baseline, we can estimate the system overhead of both models as follows: the *Basic* model that uses 13.6kB per antenna will require 8.1 times more space to store the whole database; the *Enhanc*ed model that uses 172kB per antenna will require 91.3 times more space to store the whole database.

Table 7.1 shows the result of experiment 2, which is the average execution time necessary to encrypt the antennas on mobile devices.

The results show the performance of the *Basic* and *Enhanc*ed models. Nevertheless, note that the performance of an implementation based on the Android SDK (*Basic* model) is not directly comparable to an implementation in C++/NDK (*Enhanc*ed model): the library in C++ has several optimizations that the Java counterpart has not.

The execution time for the *Enhanc*ed model around 240 [ms] in newer devices, which is inside the range of what can be considered as practical for a mobile device. However, the execution time for the *Basic* model is inadmissible for devices with a single processor, as it will periodically block the device usage for several seconds. However, given the current phone replacement rate, this shouldn't be a concern in a near future.

Figure 7.2 shows the results of experiment 3, which is the average execution time necessary to execute the count function on the server-side.

The results show that the *Basic* model out-performs the *Enhanc*ed model. On average, the *Enhanc*ed model doubles the execution time of the *Basic* model. When compared to the current implementation of Adkintun Mobile (non-encrypted version), which is not shown in the plot but takes on average 1500 [ms], the count function evaluation for both models is time consuming. However, both the *Basic* and *Enhanc*ed models can be considered as practical, depending on the usage of the system. They are not suitable for real-time analysis, but can be considered adequate for offline statistics, which can be periodically computed.

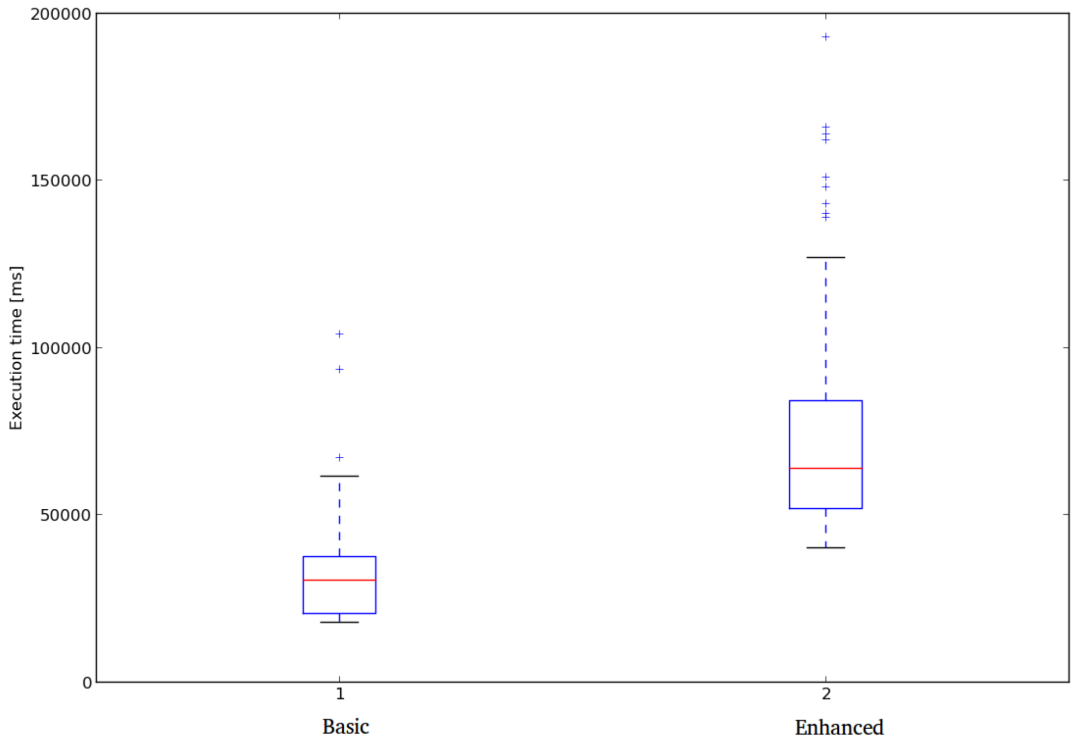

Figure 7.2: Experiment 3 - Average execution time of the count function evaluation without input size restriction.

Figure 7.3 shows the results of experiment 4, which is the average execution time necessary to perform a count function on the server side for different query sizes.

The results show that the count function behaves linearly with respect to the size of the evaluation set. This is as expected, considering that the number of operations required to perform the count function is proportional to the number of records that are taken as input.

The results also give some insights on which input sizes to chose, if we wish to parallelize the tasks to reduce the execution time of the count function evaluation. For example, if we define that an average of 10 [s] is adequate for an application using the *Enhanc*ed model, we know directly from the plot that each task should take an input size of around 300 records. The number of parallel tasks can be obtained if the total input size is known.

Finally, Figure 7.4 shows the result of experiment 5, which is the average execution time necessary to decrypt the result of a count function evaluation, on the server side.

The results show that, as with the encryption, the *Enhanc*ed model performs better than the *Basic* model, which doubles on average the execution time. However, for both models, 100-200 [ms] can be considered as negligible in comparison to the count function execution time.

In general, the results of these experiments show that, even if the models, especially the

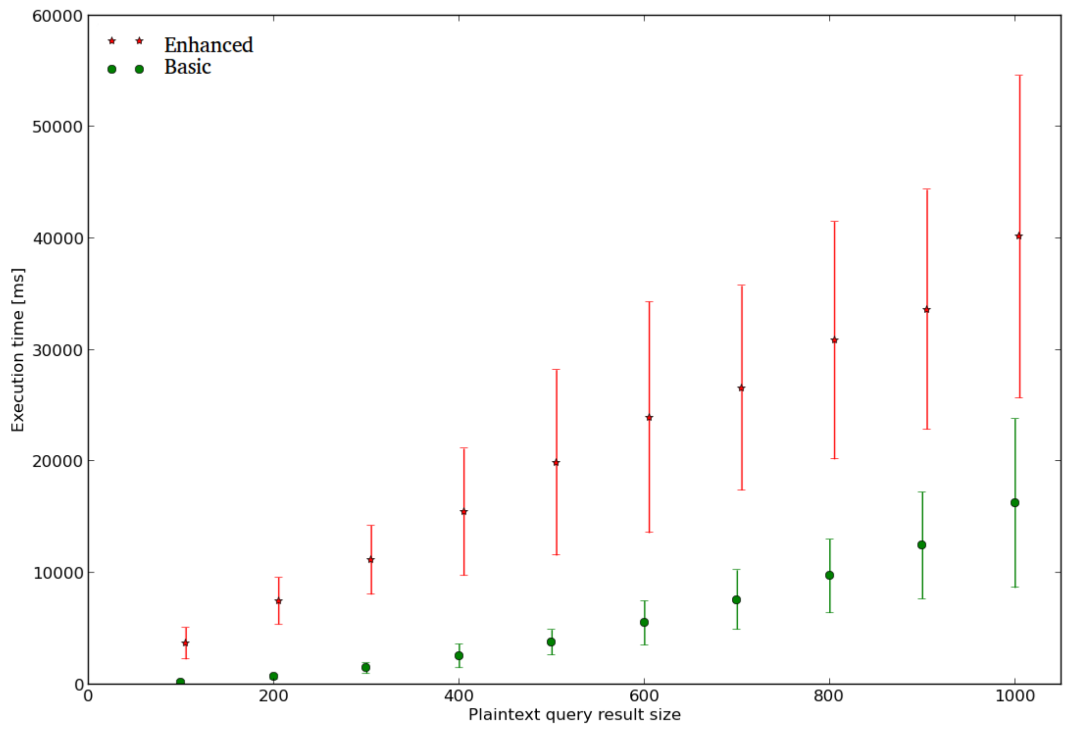

Figure 7.3: Experiment 4 - Average execution time of the count function evaluation for different input sizes.

*Enhanc*ed model, have a non-negligible overhead, its usage is in general in the bounds of a practical solution. Parallelization for the count function evaluation is an alternative to reduce the execution time, specially in the *Enhanc*ed model that follows the typical structure used in the map-reduce algorithm [12]. Finally, additional optimization work is necessary to improve the encryption algorithm used in the *Basic* model implementation, to be practical on mobile devices.

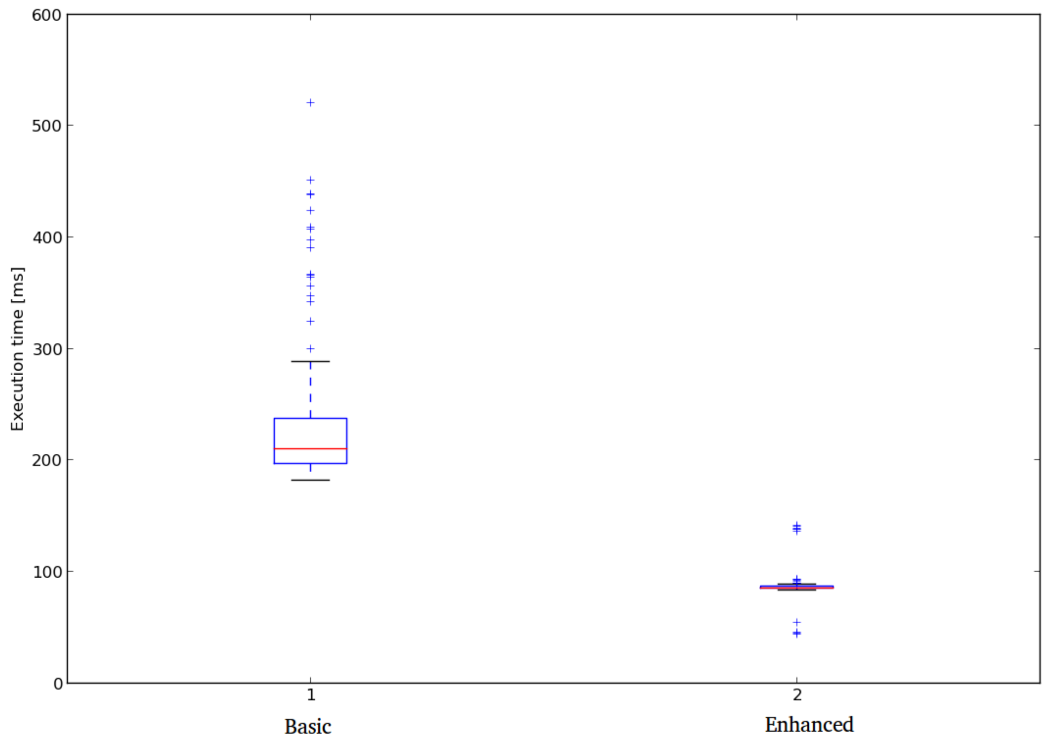

Figure 7.4: Experiment 5 - Average execution time of the decryption of the final result.

8

Discussion

This chapter discusses possible extensions of the count function evaluation. It also discusses how the trade-off between location privacy and performance of the system can be affected by these extensions.

8.1 Handling larger sets of records

In section 5.3, we discussed how p in the *Enhanc*ed model is the maximum number that can be represented by a count function, without overflow. The equivalent for the *Basic* model was also discussed. In a conservative approach, the maximum input size for the count function should then be p to avoid data corruption. However, if we wish to handle a larger input size, a less-conservative, probabilistic approach can also be taken.

With the chosen parameters in the *Enhanc*ed model, a cipher can handle a rotation, a multiplication, and up to sp additions without becoming too noisy (with $s = \frac{93}{23}$, determined experimentally). The input size of the count function can then be augmented to sp, without obtaining garbage from the decryption. We can chose this new setting, based on the following assumption:

Let S be the input set, with $|S| \leq sp$, let a^* be an antenna, then

$$Pr[count(a^*, S) \geq p] = \varepsilon, \tag{8.1}$$

with ε a negligible value.

In fact, considering that antennas are distributed in the geographical space proportional to the density of users (more users, more antennas) we can assume that the probability that a given record corresponds to a given antenna is $\frac{1}{\alpha^{\lambda+1}}$. For $|S| = sp$, the event $count(a^*, S) \geq p$ can be decomposed into the following events:

$count(a^*, S) = p$, $count(a^*, S) = p + 1$, ..., $count(a^*, S) = |S|$,

which can be enumerated as the way of choosing p records over $|S|$ records such as the p records correspond to antenna a^*; the way of choosing $p+1$ records over $|S|$ records such as the $p+1$ records correspond to antenna a^*; ...; the way of choosing $|S|$ records over $|S|$ records such as the $p+1$ records correspond to antenna a^*. As the total number of ways to assign any of the $(\alpha^{\lambda+1})$ antennas of a public set to $|S|$ records is $(\alpha^{\lambda+1})^{|S|}$, then the previously mentioned probability of equation 8.1 is:

$$\begin{aligned} Pr[count(a^*, S) \geq p] &= \frac{\binom{|S|}{p} + \binom{|S|}{p+1} + \ldots + \binom{|S|}{|S|}}{(\alpha^{\lambda+1})^{|S|}} \\ &\leq \frac{(|S| - p)\binom{|S|}{\frac{|S|}{2}}}{(\alpha^{\lambda+1})^{|S|}} \\ &\leq 2\mathrm{e}^{-150052} \end{aligned}$$

for the chosen parameters, which is in fact a negligible value.

Even larger sets of records can however be handled with an extra decryption step (i.e., adding in plain-text the partial results of each counting function performed over subsets of size $\leq sp$).

8.2 Counting users per antenna in a day

Researchers may wish to count the number of users that connected to antenna a_i during a given day. More precisely, they may wish to know the antenna load during the day, which means to compute the number of distinct users connected to the antenna for each time interval during this day.

The approach will be as follows. For each time interval, retrieve the records corresponding to the public set of antenna a_i. Evaluate the count function of antenna a_i over the set of records.

Researchers will then learn the evolution of the antenna load during this day without learning users' location.

The lengths of the time intervals in which the day is partitioned, is critical. There is a trade-off between the accuracy of the result and location privacy (but it does not have an impact on the performance of the evaluation):

- For a given day, the way this day is partitioned into intervals does not affect the total number of records corresponding to the day. For example choosing a smaller interval will augment the number of count function evaluations, but reduce the input size for each evaluation; and choosing a larger interval will reduce the number of count function evaluations but increase the input size for each evaluation. As the total number of records of the given day is not affected, and as for both models, the execution time of the count function is linear with respect to the size of the input (as explained in Section 7.2) the size of the interval will not have an impact in the performance of the whole query.

- Choosing larger intervals to partition a given day may impact the accuracy of the result. Think about a user in the border zone between two antennas. Alternately, the mobile phone will be connecting to one antenna and the other as in a *ping-pong* game. If the interval chosen is larger than the time it takes to connect and reconnect to the same antenna during a *ping-pong* event, the user will be counted twice in the same antenna, in the given interval.
- Choosing a smaller interval size to partition a given day would reduce the number of records per count function. If the interval is small enough, only a few number of users may be represented in the sample: this may augment the probability of guessing if user u_j is connected to antenna a_i from the result of the count function.

An approach to find the appropriate size of the interval would be to estimate the average change rate of users from one antenna to another.

8.3 Counting records by zone

Consider the scenario where researchers are interested in counting the records per zone instead of the records per antenna. A zone would be defined as a cluster of antennas, which means a set of antennas that share a geographical space or neighborhood. This may be useful when analyzing the quality of access to mobile Internet in a city area.

We will show how the efficiency of this task under the *Enhanc*ed model can be improved, whereas the *Basic* model behaves poorly. We will also show the trade-off between efficiency and location privacy.

The approach is as follows. Antennas will are assigned to public sets in a slightly different way. Instead of assigning them randomly, we will assign antennas first to clusters of antennas or zones of size α (where α corresponds to the number of antennas per private zone in the *Enhanc*ed model). Then α zones will be randomly picked and assigned to a public set, as shown in Figure 8.1.

In the *Enhanc*ed model, zones are still managed as private zones, which means they are stored encrypted on the database. In the *Basic* model, we will have $\alpha_{basic} = \alpha^2$ antennas per public set, without disclosing either the membership of a record to a particular zone.

The counting function for zone $Z_j \in S_k$, where S_k is the k^{th} public set, will be evaluated as follows:

- *Basic* model:

  For each antenna a_i belonging to Z_j, evaluate the count function for antenna a_i over

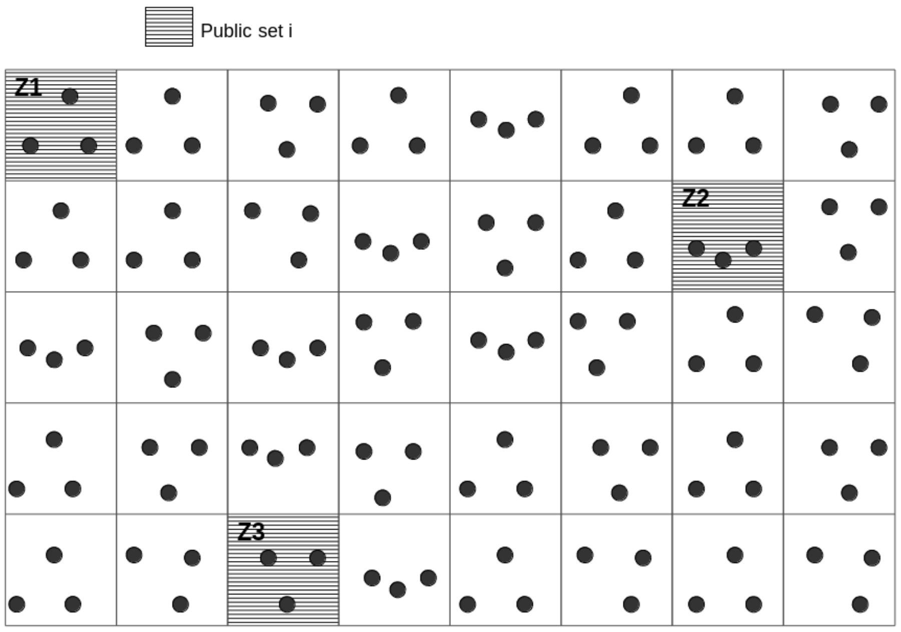

Figure 8.1: Public sets formed by randomly selecting clusters of antennas.

the records that belong to S_k.

- *Enhanced* model:

  Add up the private zone ciphers that belong to S_k. Return the decryption of the j^{th} slot.

If $T_{basic}(C)$ is the average execution time of the count function over an antenna in the *Basic* model, it is direct to deduce that the execution time for the count function over a zone will be $\alpha T_{basic}(C)$, where α is the number of antennas in the private zone Z_j.

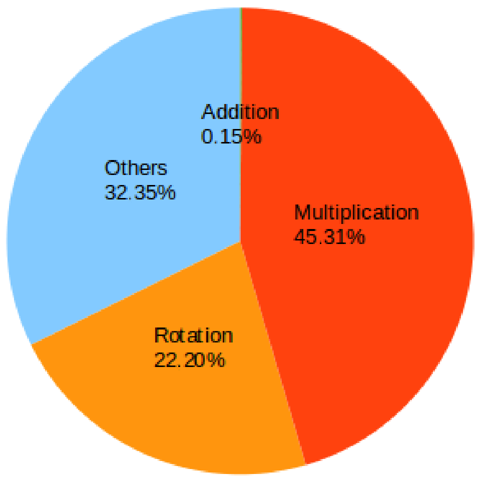

Figure 8.2: Distribution of the execution time by task, for a count function in the *Enhanced* model.

For the *Enhanced* model the intuition is that the execution time should be less than the execution time of a standard count function. This is because only addition is required, avoiding the rotation and multiplication steps of the standard count function. To measure the improvement in performance, Figure 8.2 shows the distribution of the execution time by task for a count function in the *Enhanc*ed model. It shows that a count by zone function should take on average only 32.5% of the execution time of a standard count function evaluation.

For the particular model parameters we are using, and for a count zone function evaluated over a set of 1000 records, the *Basic* model should take on average $\alpha T_{basic}(C) = 41 \times 26[s] = 1066[s]$, while the *Enhanc*ed model should reduce the execution time to $0.325 T_{enhanced}(C) = 0.325 \times 40[s] = 13[s]$. The numbers used in the estimation are taken from the results presented in Section 7.2.

Regarding the location privacy trade-off, we may ask if the new assignation of antennas into clusters discloses more user's location information.

On the one hand, the sparsity of antennas in a public set is reduced by grouping subsets of antennas. From this point of view, the probability of guessing if one record belongs to a cluster is $\frac{1}{\alpha}$.

On the other hand, clusters of antennas occupy a wider area than a single antenna, increasing the obfuscation of a record. Let δ be a parameter related to the diameter of the area occupied by the cluster. Then, the probability of guessing if one record belongs to a particular location will be reduced to $\frac{1}{\delta\alpha}$.

9

Conclusion

This chapter summarizes the principal results and contributions of our work, and proposes future work on the topic.

9.1 Results and contributions

The present work shows that our model provides location privacy while allowing the efficient computation of a predefined set of count functions. In particular:

- Our security model preserves users' location privacy in the different modules of the system, by ensuring collector-processor obliviousness and decryptor obliviousness (Section 4.2), based on the semantic security of the Paillier cryptosystem and the BGV cryptosystem.
- Our model presents a non-negligible overhead proper of the trade-off between location privacy and system performance. However, the overhead is reasonable, and the implementation shows an appropriate performance of the count function evaluation in an offline data analysis context (Section 7.2).
- The *Basic* model outperforms the *Enhanced* model in functions such as counting records by antenna, taking on average half of the execution time (Section 7.2); however, the *Enhanced* model is suitable for functions such as counting the records by zone: the execution time should be reduced to 32.5%, while in the *Basic* model, the execution time should be increased by a factor equal to the number of antennas in the zone (Section 8.3).

The contributions of this work are to propose a practical use of FHE for location privacy, that includes implementations of modules on both server-side and mobile devices; and provides a comparative evaluation of the system performance and location privacy for implementations using FHE and PHE.

9.2 Future work

Although execution time of count function is practical for non real-time applications, researchers may need to perform this kind of analysis for a large set of antennas. It would be an interesting topic to study how the performance can be improved by means of parallelism.

Another interesting problem is key management. Systems usually update keys periodically for security reasons. This means that the database will have historical records under different keys, which means that they are not compatible under homomorphic operations. Studying how to build a module that will transparently manage the computation over records with different keys should be studied further.

In this work we focused on simple aggregation queries. Future work should study how to implement more complex functions, such as users flows between antennas.

Finally, as battery usage is a critical topic in current smart-phones, it would be interesting to evaluate the power consumption of the module running on mobile devices. If the energy consumed by the application is considered as a currency, how much is the user willing to pay to volunteer in the Adkintun Mobile project, and how much more is he willing to pay to preserve his location privacy?

10

Appendix

10.1 Indistinguishability under chosen plaintext attacks (IND-CPA)

The IND-CPA property is represented as a game. An adversary A plays the game. A has access to invoke an oracle that takes as input to messages m_0 and m_1 and returns c, the encryption of one of them. A has to guess which of the two messages was encrypted.

Given two plaintext messages m_0 and m_1, $|m_0| = |m_1|$, the goal of A is to guess if c was generated from the encryption of m_0 or from the encryption of m_1. In the game, the pair (m_0, m_1) is submitted by A as input to an oracle. The oracle flips a coin $b \in \{0, 1\}$. If $b = 0$, the oracle returns $c \leftarrow encrypt(m_0)$. If $b = 1$, the oracle returns $c \leftarrow encrypt(m_1)$. A guesses the value of b and succeeds in the game if the guess is correct.

In a semantically secure cryptosystem, the probability that A guesses correctly is $Pr \approx \frac{1}{2}$. This means that A is not able to distinguish between the encryption of a message m_0 and the encryption of a message m_1 (of the same length), and that his advantage in the game is negligible.

11

Bibliography

[1] Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 376–385. IEEE, 2008.

[2] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *Proceedings of the twentieth annual ACM symposium on Principles of Distributed Computing*, pages 274–283. ACM, 2001.

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

[4] Andrew J. Blumberg and Peter Eckersley. On locational privacy, and how to avoid losing it forever. `https://www.eff.org/files/eff-locational-privacy.pdf`, August 2009. Electronic Frontier Foundation.

[5] Francesco Bonchi. Privacy preserving publication of moving object data. In *Privacy in location-based applications: research issues and emerging trends*, volume 5599, pages 190–215. Springer, 2009.

[6] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

[7] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 97–106. IEEE, 2011.

[8] Javier Bustos-Jiménez, Gabriel Del Canto, Sebastián Pereira, Felipe Lalanne, José Piquer, Gabriel Hourton, Alfredo Cádiz, and Victor Ramiro. How adkintunmobile measured the world. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 1457–1462. ACM, 2013.

[9] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology–CRYPTO 2011*, pages 487–504. Springer, 2011.

[10] Ivan Damgård, Jens Groth, and Gorm Salomonsen. The theory and implementation of an electronic voting system. In *Secure Electronic Voting*, pages 77–99. Springer, 2003.

[11] 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.

[12] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[13] Matt Duckham and Lars Kulik. Location privacy and location-aware computing. *Dynamic & mobile GIS: investigating change in space and time*, 3:35–51, 2006.

[14] European Union Agency for Network and Information Security (ENISA). Algorithms, key size and parameters report. `http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014`, November 2014.

[15] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[16] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *Advances in Cryptology–EUROCRYPT 2011*, pages 129–148. Springer, 2011.

[17] Craig Gentry, Shai Halevi, and Nigel P Smart. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology–EUROCRYPT 2012*, pages 465–482. Springer, 2012.

[18] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. Secure multiparty aggregation with differential privacy: A comparative study. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, EDBT '13, pages 155–163, New York, NY, USA, 2013. ACM.

[19] Shai Halevi and Victor Shoup. Design and implementation of a homomorphic-encryption library. IBM Research, 2012. `https://github.com/shaih/HElib`.

[20] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[21] Moon Sung Lee. On the sparse subset sum problem from Gentry-Halevi's implementation of fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011:567, 2011.

[22] Iraklis Leontiadis, Kaoutar Elkhiyaoui, and Refik Molva. Private and dynamic time-series data aggregation with trust relaxation. *IACR Cryptology ePrint Archive*, 2014:256, 2014.

[23] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology—EUROCRYPT'99*, pages 223–238. Springer, 1999.

[24] Raluca A Popa, Hari Balakrishnan, and Andrew J Blumberg. Vpriv: Protecting privacy in location-based vehicular services. In *USENIX security symposium*, pages 335–350, 2009.

[25] Ramesh Rajagopalan and Pramod K Varshney. Data-aggregation techniques in sensor networks: a survey. *IEEE Communications Surveys & Tutorials*, 8(4):48–63, 2006.

[26] IBM Research. IBM public challenge for fully homomorphic encryption. IBM Research, 2011. `http://researcher.watson.ibm.com/researcher/view_project.php?id=1548`.

[27] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[28] Elaine Shi, T-H Hubert Chan, Eleanor G Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *NDSS*. The Internet Society, 2011.

[29] Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography–PKC 2010*, pages 420–443. Springer, 2010.

[30] Latanya Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, October 2002.