Design and Evaluation of a Location Based Encryption Protocol

Master Thesis

submitted by

Ahmet Eroglu

at the

Universität Konstanz

Faculty of Sciences

Department of Computer and Information Science

1. Evaluated by: Prof. Dr. Marcel Waldvogel
2. Evaluated by: Prof. Dr. Sven Kosub

Konstanz 2019

Abstract

Over the past decade, Location Based Systems (LBSs) have been becoming more and more popular among users, as these services help them to locate their acquaintances, share their favorite places, find the nearest hospital or even find their beloved ones. These services, however, are usually built from the ground up with only practicality in mind. In general, security countermeasures in LBSs are aiming at different aspects of privacy, such as encryption of certain user data and/or providing secure communication between the users and the server during data transmission. However, it appears that direct or inadvertent disclosure and/or misuse of location information of the users gives rise to certain security threats. For example, when the location privacy is lost, confidential information regarding people's private lives can easily be obtained. Work or home addresses, in addition to certain places relevant to religious views can be revealed. Places that are frequented based on income levels may also be disclosed. Accordingly, this information may potentially be used for several malicious purposes, such as stalking a person, exploitation of one's private life, taking advantage of users based on the places they frequent and even threatening them. Hence, new countermeasures against the loss of location privacy should be presented to mitigate this very problem, as today's LBSs are operating on a highly location-pervasive way.

This master thesis addresses the uprising problem of location privacy and thereafter introduces a new location-based encryption protocol called LOCKUM. We show that such a protocol is highly effective in retaining the location privacy for certain use-cases, such as location-based messaging applications, location-based games and location-based advertisement services. LOCKUM provides a scheme where users can send each other messages that are geographically locked to a certain area by encrypting the messages using the location data; this way, users are given the opportunity to communicate using their location information while still retaining their location privacy. Current encryption techniques don't facilitate using location data directly, and we demonstrate that location data are, in and of itself, not suitable to be used directly as an encryption key due to its low entropy, but rather it should be a part of the key construction process. Hence, we show how location data itself could be used to encrypt information while still retaining the location privacy. Lastly, after the protocol is introduced, evaluation of the protocol and an improved version are presented.

Contents

1 Introduction

Recent advances in the smartphone industry have led to the emergence of a new era by changing the way users utilize the internet. Being able to access information in the blink of an eye has opened new avenues to users, companies, and application developers. Social networking, online shopping and instant messaging are just a few examples which constitute this tremendous internet starvation. Under the circumstances, mobile applications cater to this era at an unprecedented pace, in that we can accomplish almost anything online by a few touches on these nearly bezel-less, vividly-looking phones through these apps.

As technology advances and devices are equipped with various sensors including receivers, such as Global Positioning System (GPS), one of the emerging services is Location Based Systems (LBSs). In LBSs, the users' location is captured, actively tracked and even stored so as to deliver a service, such as social networking, locating the nearest restaurant or making a short distance trip with Uber[1] service [18]. As described in [20], some of the most popular LBSs can be shown as:

- **Location Based Social Networks (LBSNs)** Social networks and instant-messaging applications are the common examples of LBSs. Through these systems, not only we are able to communicate, but also we can add geo-tags to our photos, share our location instantly to notify our friends, learn their whereabouts, or simply see the events happening nearby to join. Hence, such services actively capture, use and store the location of their users so as to provide a service.

- **Location Based Gaming (LBG)** Essentially, the gaming industry has gained a new perspective with the help of Location Based Gaming (LBG). Games like Pokemon GO[2] or some other augmented reality games are the epitome of location-based games. These games are mostly based on overlaying the virtual world onto the real world to engage users. Location information here plays a tempting role to engage users, as in allowing them to earn trophies on certain spots. As these games are getting more and more popular, it has become very common among users to share their location data without any precaution.

- **Location Based Advertisement (LBA)** In today's world, companies are using different strategies to attract customers so that they can increase their revenues. For instance, using location-based advertisement services, discount coupons are distributed with the help of mobile applications. Alternatively, Bluetooth beacons are used to detect nearby people. Through these beacons, users are asked to visit close-by shops to benefit from discounts. In such systems, users location is actively tracked and used. While doing that, the users' location privacy is often not taken into consideration, as the sole purpose is to increase the profit of a company.

Even though LBSs have brought new dimensions to our everyday life, many problems have arisen from the security and privacy standpoints. Existing privacy solutions in many of the other LBSs are heavily relying on encryption methods to protect identities of the users or the corresponding user details during the user-user and user-server communication. However, the number of services are few to none when it comes to retaining location privacy whilst providing the same functionality for the users. All in all, disclosure of location information wasn't a crucial threat factor, as it offered a lot of practicality for a long time. However, with the dazzling increase in the usage of LBSs causing to track our location continually, location privacy became important and started to raise concerns for all users[3]. Like any type of privacy, location privacy should also be under the control of the users. In the circumstances, breach of a person's location privacy will wheel out certain consequences. Let's take a closer look at possible outcomes of the loss of location privacy:

[1] more information at https://www.uber.com/ch/en/

[2] more information at https://www.pokemongo.com/en-us/

- Disclosure of a person's home and/or work address
- Learning people's habits based on the places they visit
- Revealing the religious beliefs of individuals, visits to churches or mosques
- Exposing the sexual diseases of the users if they visit a hospital for certain treatment.
- Giving away people's income levels based on the places they visit
- Abusing people with advertisements, depending on their proximity

As can be seen from the above factors, the loss of location privacy constitutes a direct threat when misused. In today's society, it is highly natural and expected that location information has to be exchanged between different entities. For example, people need to inform authorities to state their residency, cannot get a credit card without sharing their address or simply share their upcoming holiday plans with their friends. In such a relational context, many entities play a crucial role. Currently deployed LBSs are operating on a highly location-privacy invasive manner. To accommodate their services, service providers and third parties are constantly learning the location information of the users. Therefore, it is essential to infer that location privacy is precious and inevitable and, consequently, should not be subject to dissemination, collection, and processing without required permissions.

1.1 Goal of the thesis

In this master's thesis, we focus on location privacy in certain LBSs, such as location-based instant-messaging applications, location-based games, and location-based advertisement systems. In many of these LBSs, the location information is used in a pervasive way. As mobile applications pose a great challenge in this manner, users are asked to prove their location information regularly.

We aim to present a new location-based encryption protocol called LOCKUM to confine messages to a certain area by encrypting these messages using location information during communication between the users while still allowing them to retain their location privacy. As we mentioned before, dissemination of location privacy creates certain consequences that might possess a direct threat for the users. Hence, it is very important to provide new solutions to mitigate the problem of dissemination of location privacy. Our protocol targets these LBSs to protect the location privacy of the users. After the introduction of LOCKUM, we further analyse and evaluate the protocol from security and practicability standpoints. Later, we present possible improvements over the proposed protocol to strengthen LOCKUM's security.

1.2 Requirements

Location-based encryption refers to encryption of messages using location data, which consist of latitude and longitude, so that the ones who can decrypt the encrypted messages are those who are currently positioned in this predefined location area decided by the sender[31, 25, 16]. This often takes place with the help of a smartphone, a laptop or a similar device that is capable of fetching its current coordinates. Usage of location data in connection with existing encryption protocols provides an additional dimension to location privacy [31]. When this additional dimension is added into current encryption schemes, it is assured that information will be able to be read only by devices which are verified to be in this particular area.

When designing such an encryption protocol, which aims at confining the information in a specific area, we deduced that six properties should be ensured by a good and secure location-based encryption protocol, defined similary in [6]:

1. A message should be decryptable, if and only if the user is present at the location.
2. A message should be decryptable only in one particular location area where it's confined.
3. From the practicability and scalability standpoints, no additional hardware or equipment needs to be installed.
4. A user should not be able to read an encrypted message for the second time without being there if he receives a message at the same location.
5. Users' location should never be revealed or tampered with, e.g. to a server or any other intermediaries.
6. Users should not be able to cheat their whereabouts to read a message. E.g. using a fake GPS

As can be seen from the above properties, many factors play a role in building such a system. For example, users can cheat their whereabouts; thus, only relying on the encryption methods would not be enough. In such cases, the origin of the location data should be further checked using different methods. However, additional components used to achieve this extra control might also jeopardize the secrecy of location information, such as using hardware-based components to verify location claims of the users. Accordingly, a secure system should provide the required functionality while using the minimum number of the components, as each additional component would mean bigger overhead and new maintainability problems.

1.3 Use cases

In this master's thesis, we are mainly focusing on three use cases which are LBSs providing different types of services based on users' location information: location-based messaging services, location-based gaming services, and location-based advertisement services. As our protocol is not designed to encrypt highly sensitive information, these location-based systems fit in our protocol design very well.

Location-based Messaging: Some of the current instant-messaging applications allow their users to send messages that are geographically locked to a certain area to provide a fun, new way of communication. In such applications, a user leaves a message in a certain area for another user. Afterward, the receiver gets a notification and has to go to this specific area to read his message. However, many of these applications do not provide such functionality by taking users' location privacy into consideration, as they actively capture, store and use location information of the users with only practicality in mind. Accordingly, users' location privacy is destroyed due to this pervasive way of handling the location data.

Using our location-based encryption protocol, such location-based messaging services can allow their users to retain their location privacy while still providing the same functionality. In Figure 1, an example situation is illustrated to show how a location-based encryption protocol would function using the location data. As can be seen from Figure 1, a sender decides on a certain area where he wants to encrypt the message. He uses latitude and longitude of this area as inputs to location-based encryption protocol. After the message is encrypted, the sender sends the message using End-to-end Encryption (E2EE). This way, the encrypted text is prevented from tampering and only decryptable at the receiver's side. Once the receiver gets the encrypted text, he starts to search the correct location by using different locations captured by his phone. If he goes to the correct place, the encrypted message is successfully decrypted, and he can read the message. In summary, unlike the traditional location-based messaging applications which are operating on a highly location-pervasive way, our protocol protects users from revealing their location privacy while still allowing them to have the same functionality offered in other location-based messaging systems where confidentiality of location information is ignored.

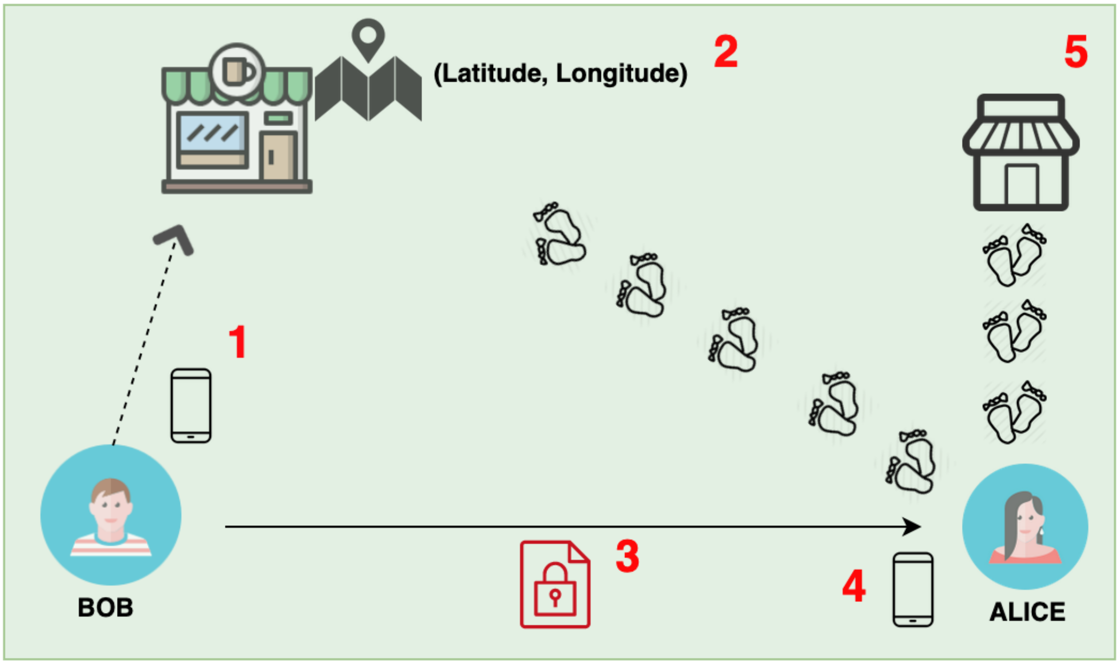

Figure 1: Sender Bob uses his smartphone (1) to fetch the location of a cafe (2) where he wants a message to be encrypted. He uses location-based encryption protocol and encrypts the message using latitude and longitude of the cafe (2). Later, using the secure channel between the Bob and Alice, he sends the encrypted text (3) to Alice. Alice, then, uses her smartphone (4) and goes to different places like (5) to decrypt the encrypted text (3). If she goes to right location, the cafe in our case (2), she can successfully decrypt the encrypted text (3) and read the message.

Location-based Gaming: The number of mobile phone subscriptions worldwide had reached almost 7 billion in 2013[8]. As the devices are highly widespread and capabilities of them are enormously increasing, advance gaming is no longer such a fantasy in today's world. Many mobile devices nowadays are also portable gaming tools. The delightful fact about gaming is that they keep the users occupied with challenges, fun, and some other factors which make them feel ambitious. Games like Pokemon-Go[3] and Ingress[4] are the epitomes of such location-based gaming services. Indeed, to play such location-based games, users are often asked to give their consent with regards to sharing their certain confidential data, such as location information. Upon getting users' consent, game providers allow users to use their entertainment services. However, this careless act done by the users allows game providers to store and use such confidential data very easily. Under the circumstances, everyday locations of the users including the work and home address data are just some of the information which is known to game providers although revealing such confidential data should not be so easy. This being the case, location-based encryption would be used as a part of the games in various meanings. Different types of trophies, prizes or bonus items might be encrypted with a specific location, incentivising users to go there to get their bonus by challenging them with some triggering factors, such as fun and rewarding feelings. Usage of such location-based encryption protocol would prevent players from revealing their location to any other entity while still being engaged in a fun experience.

[3] more information at https://www.pokemongo.com/en-us/
[4] more information at https://www.ingress.com/

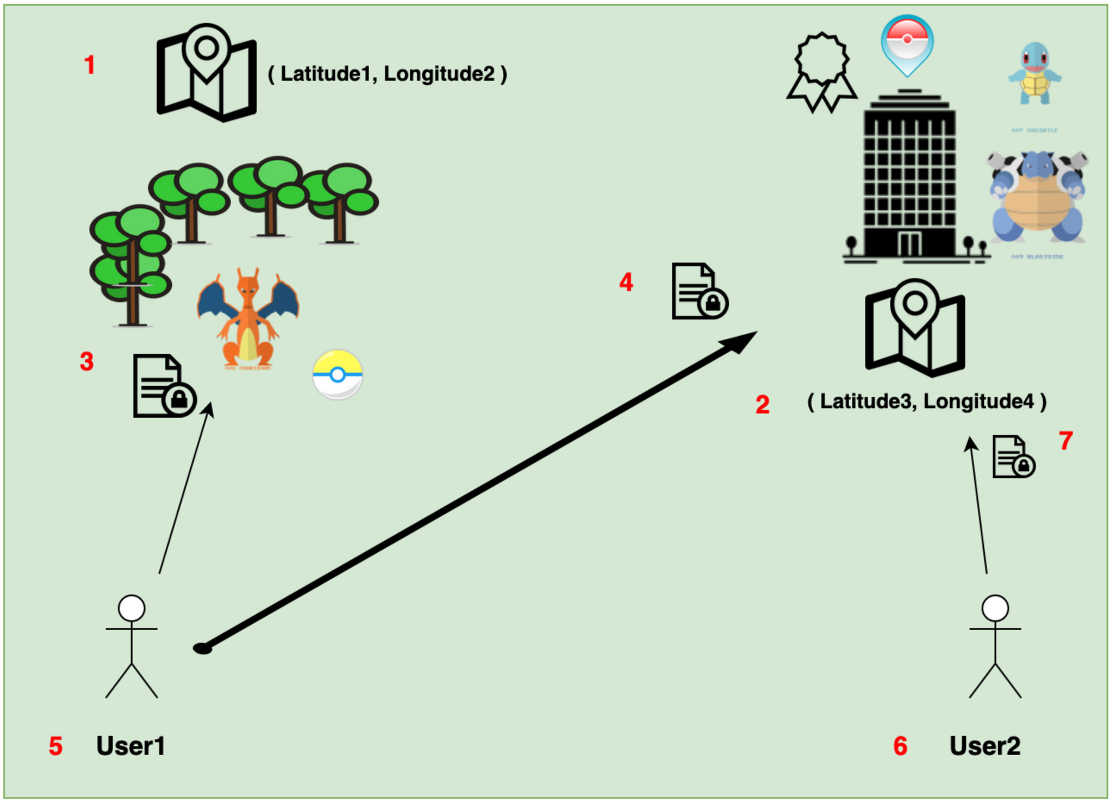

Figure 2: In this example, User1 (5) gets a cipher-text (3) which is encrypted with the location of a certain park (1) and contains a prize depending on the game context from another user sent by E2EE. He must go to this area (1) in order to decrypt the cipher-text (3) and get his prize. Similarly, user1 (5) can encrypt bonus items for User2 (6) using location of another place in some area (2) and create an encrypted-text (4) out of it which contains game related prizes and trophies. Afterwards, User1 (5) sends this encrypted-text to the User2 (6) via E2EE. Later, User2 (6) should go to this place (2) in order to decrypt the encrypted-text (7) which is sent by User1 (5).

In Figure 2, how location-based encryption protocol could be used in location-based games is illustrated. As can be seen from Figure 2, players' location privacy is protected as all data are sent using a secure channel thanks to E2EE. This way, users could still engage in the game while protecting their location privacy, as no data are revealed to any other entity.

Location-Based Advertisement Services: LBSs are also used by companies in the shopping industry. For example, Bluetooth beacons and similar solutions are used to send notifications to nearby people based on their proximity. In most of these cases, location information of the customers might be subject to dissemination, as only practical purposes are taken into consideration by such corporations. A shop which wants to increase their sales numbers would be even more appealing to customers by encrypting discount coupons using the shop's location and sending them to both online and regular customers.

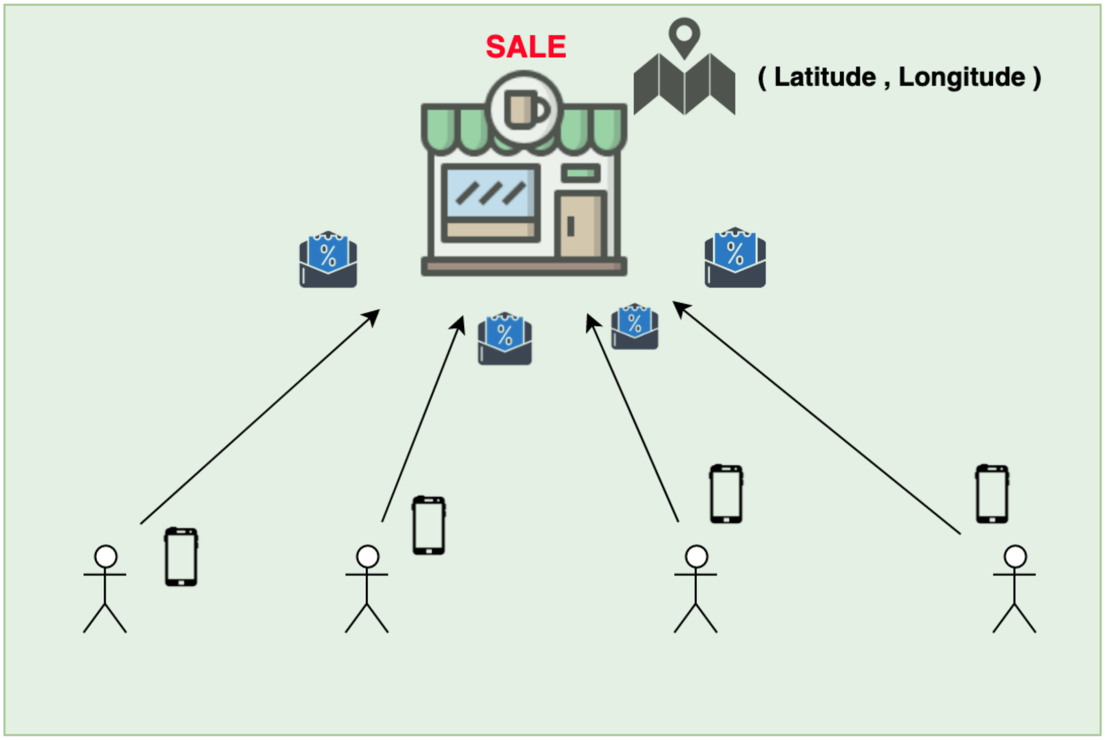

Figure 3: A local shop creates many discount coupons for their customers. These coupons are later encrypted with the location of the shop and sent to the customers' phones using E2EE. Accordingly, customers should visit the shop in order to decrypt the encrypted discount coupons. Once the coupons are decrypted, they can be used for in-shop and online purchases.

Discount coupons would be then used to make purchases both online and in the shops. This way, customers would have to come to get their discount coupons using their smartphones to benefit from this chance. In Figure 3, an example case is illustrated where a shop distributes encrypted discount coupons to their customers rather than using traditional ways. In such a scenario, handing out discount coupons, say a coupon code printed on a paper, would be much more expensive. By using such an online approach, users' location privacy is still retained while allowing them to benefit from several opportunities in today's paperless world.

1.4 Assumptions

In this section, we will present what assumptions are made to facilitate our protocol's practicability. Justifications of the assumptions heavily rely on fulfilling the requirements stated in section 1.2.

- **End-to-End Encryption**:

  Since late 2015, E2EE became a de facto standard in most of the instant messaging applications (e.g.,WhatsApp, Facebook Messenger, Viber, LINE, and KakaoTalk)[13]. The terms constituting the endpoints in a given communication are well-discussed in [10]. In such services, the communication of the two devices requires a medium to deliver messages from one device to another by way of Short

Message Service (SMS), General Packet Radio Service (GPRS) Signal or Wireless Fidelity (WIFI). Now, when communication takes place, in all of these circumstances, there is mainly going to be an intermediary server which handles the transmission, storing and routing of the messages. Although the communication between the server and the clients are claimed to be encrypted and so-called "secured" through a channel, that doesn't obstruct servers from being able to view the content of the messages sent during the storage and routing. Using public-key cryptography, E2EE, in principle, prevents any intermediaries between the sender and the receivers from tampering with the message. Hypothetically, this should conclude the questions regarding basic privacy concerns during communication; however, in [19], Amir and Hemi show that this communication mechanism is still subject to improvement and the term E2EE itself might be used deceptively way to make users feel secure. In practice, we refer to disabling anyone in-between the sender and the receiver from viewing the context of the message, therefore, fulfilling the ultimate protection between the recipients using E2EE.

As defined in section 1.2, a good location-based encryption protocol should retain location privacy and none of the location-related information, as well as auxiliary information that is used in this process, should not be revealed or tampered with other than by its recipients, including the servers. To assure this, we require an E2EE communication mechanism between the users so that none of the information that assists to decryption is known to anyone else other than the recipients during the transmissions.

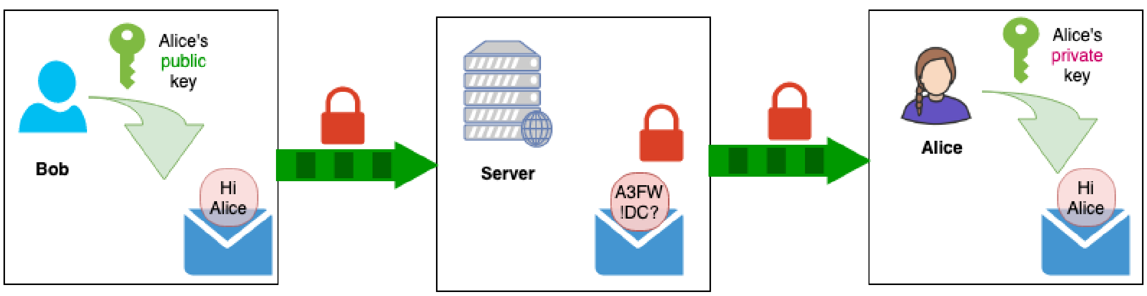

Figure 4: In E2EE, Bob sends a message to Alice using her public key, which can be decrypted only by the corresponding private-key that is only known to the receiver, in essence, this creates a so-called "secure" channel between these two. Thereafter, Alice uses her own private key to decrypt the message. Hence, any intermediaries are prevented from tampering with the information.

In Figure 4, an insight into E2EE is given. As can be seen from the figure, two mathematically-related keys play a role while providing such functionality. If a message is encrypted using someone's public key, the only way to decrypt the encrypted text will be to use the corresponding private key. However, other intermediaries won't have the corresponding private key, as this key is only possessed by the receiver. Accordingly, the intermediaries won't be able to tamper with the encrypted-text.

- **Trusted Users**:

  One of the biggest challenges when designing an LBS is creating a scheme that is resistant to fraud. When we think about an LBS, two possible suspects who can tamper with the system: an outsider or an insider. Specifically, as an insider- if we consider a recipient who gets a message and is in the process of reading a message without being at the certain location might do so by faking his location from the very beginning.

  Several approaches have been developed over the course of the last decade to overcome this verification of location information so that malicious purposes are achieved. Hence, when designing a location-based encryption protocol, this leaves the scientists and developers with nothing but a binary

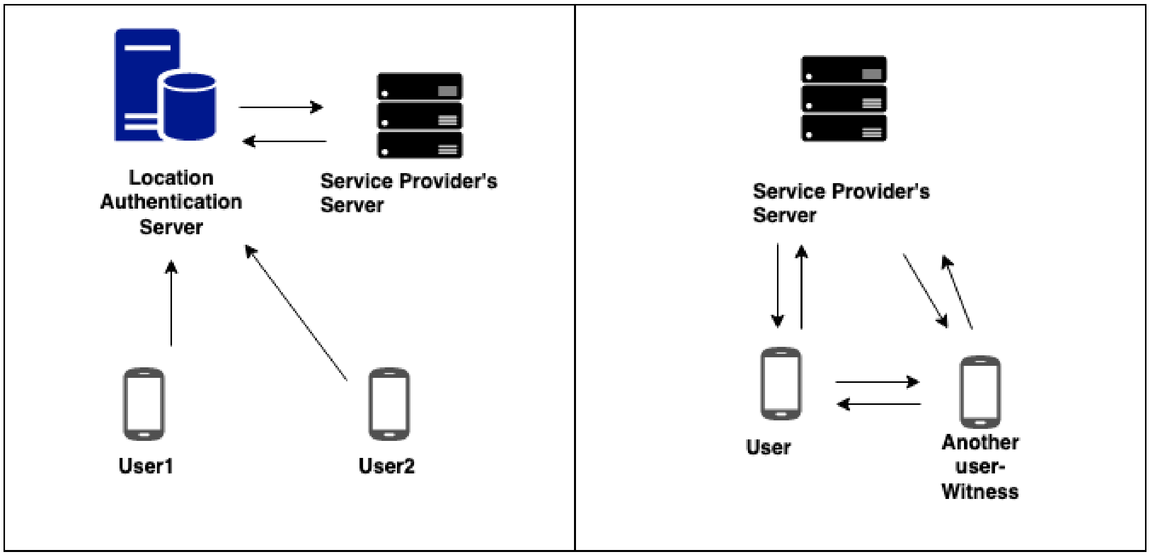

Figure 5: Left hand side of the figure shows how an additional server is used as a location verification point between the users and the server. Right side of the figure shows a scheme where users collaborate with each other to be a witness for their location claims.

decision of whether or not a third party (e.g. another server, user, device) should be introduced into the system so that the users' claimed location would be verified. Figure 5 illustrates these two possible approaches which are mainly used in these systems.

In these circumstances, we believe that introducing a third party to verify users' location contradicts with the requirements, which we defined before, as location information shouldn't be known by any other entity other than its recipients. Therefore, we assume that our users are trustworthy in the sense that they do not aim at faking their whereabouts. Although we define our users as trusted, this very question remains as the main drawback of almost all location-based encryption protocols depending on their design and implementation, which might lead to the fact that users might fake their location and misuse the service. However, in the improved version of the protocol which we present later, we introduce two countermeasures against this problem and show how it could be assured that the faking mechanism can be prevented while the current version of the protocol is prone to this very problem.

1.5 Thesis contribution

The main contribution of this master thesis is designing and evaluating a new location-based encryption protocol called LOCKUM. How location data can be used as an additional layer of security while users still don't give up on their privacy requirements has been presented in this thesis. In this way, we have shown a way in which location information is used as a part of the system to construct a cryptographically secure encryption key so that the information can be encrypted. Location-based services that provide functionality by actively capturing the location information can fortify their users' privacy by embodying such an approach. A previous study [25] introduced LDEA (Location-Dependent Data Encryption Algorithm) which creates an encryption key using location data. Later, [38, 16] proposed their own version of the encryption scheme which also included the speed of the users and a fuzzy vault approach along with location data. LOCKUM is influenced by LDEA in a unique manner and in our work, we not only strengthen the key construction process mentioned in the above studies but also provide an implementable protocol design from the ground up. For example, in LDEA, the encryption key is constructed with the help of bit-wise XOR operation. Such an approach is very troublesome, as different location values might produce the same result when they are applied by bit-wise XOR operation. LOCKUM, on the other hand, uses cryptographic building blocks and takes advantage of the cryptographic key derivation functions in order to create a cryptographically secure encryption key. Nonetheless, our protocol lacks in the location verification process, as the origin of the location claims made by users is not verified. Finally, as a second contribution, the improved version of the protocol, in which we propose two possible improvements over the old version to diminish this verification problem, has been presented.

2 Related Work

A great deal of work honed in on location-based, i.e., location-dependent encryption [32, 16, 17, 1]. Besides, verification of location claims to thwart cheating in location-based encryption and location-based systems have been well-studied [36, 21, 29, 30].

In [31], location-dependent Geo-Encryption algorithm was introduced to provide an additional security layer based on location information in the area of digital media distribution, forensic analysis, in case of piracy, and encryption of military-related information with the help of non-spoofable GPS receiver. Their approach uses a PVT(Position-velocity-time) mapping function based on the receiver's expected position, speed and the time. The composed key, derived from PVT mapping function, is called Geo-lock key. This key is later bit-wise XOR-ed by a randomly generated session key with which the actual data is encrypted to compose a Geo-locked session key. The Geo-locked session key is later transmitted to the receiver. The receiver side, on the other hand, uses a non-spoofable Global Positioning System (GPS) to calculate its own PVT values using the very same PVT mapping function to create its own Geo-lock key. Thereby, this key is performed bit-wise XOR operation with the received key from the sender to get a session key, which is then used to decrypt the information, if correctly constructed. The bottleneck of this proposal is the Geo-locked session key which is based on PVT mapping function and used by the receiver. If this mapping function is somehow retrieved or fraud-ed by the receiver, spoofing would be trivial to accomplish. During decryption, it also lacks in key verification. For example, if the receiver uses his own PVT mapping function values to construct a key, there is no mechanism available to check if this was the key that leads to the correct session key for the decryption of data, as another key might result in producing a gibberish text from a given cipher-text.

Another study introduced LDEA (Location-dependant Encryption Algorithm)[25] . The main idea behind the algorithm was to include latitude and longitude information to compose a cryptographically secure encryption key. To do so, latitude and longitude information are separately calculated and then bit-wise XOR-ed with each other before getting hashed. This hash is, afterward, halved, and each half is bit-wise XOR-ed with a newly generated random key to create the final secret key which is the initial value of Data Encryption Standart (DES) algorithm. This approach is troublesome because any other two latitudes and longitudes value might yield the same XOR resultant before getting hashed. Besides, to allow some error, a tolerance-distance mechanism is provided in the algorithm; however, this causes some of the messages to be not decrypt-able even if the receiver is within the given tolerance distance range, based on our observations. Thereby, it creates false-positive and false-negative results in some of the edge cases[5]. Similarly, in [16], the dynamic speed of the receiver was included in the system to make location-dependent algorithm non-static to encrypt Multimedia Messaging Service (MMS) messages. It does this by registering a set of the coordinates and the velocity of the receiver during his movements and estimating the next position, as well as adjusting the Dynamic tolerance distance(DTD). A newer version of this approach was also presented at [26], as asymmetric encryption of the data creates a high computing load for the mobile devices.

Additionally, a way of encrypting the user sensitive folders and files on a laptop in a given specific environment, e.g. home, office, called Mobile User Location-specific Encryption(MULE) was presented in [33]. In contrast to the aforementioned studies, MULE handles the location encryption based on an initial input which is provided by a Trusted Location Device given in a certain environment. Later, this input is used to derive a secret key to accomplish encryption. Hence, it requires all the existing entities in a given environment to be in connection with the Trusted Location Device. Overall, this protocol provides an encryption scheme on the surface by encrypting the files and folders rather than the actual data content. In the circumstances, if an encrypted folder is stolen by an outsider, it will be up to fraud to decrypt this folder loaded with sensitive files. From the viewpoint of implementation, this protocol shows good insight

[5] see section 3.4.4 for detailed explanation

into a system where encryption based on a location-dependent data is used in a real-life scenario. However, it requires a trusted device which is always an extra burden to maintain against multiple attacks although they claim that using a trusted location device(TLD) is not problematic. Besides [33], [38] shows how fuzzy vault scheme could be used in connection with location information to create and implement a fully working location-based encryption protocol using an android phone.

Above-named studies specifically focus on location-dependent encryption. Secure verification of location claims is as much important and has also been focused, as it is crucial for Location Based Systems (LBSs) and location-dependent encryption algorithms [14, 22, 27, 5].If verification of the location claims is not correctly checked, the whole encryption would be eliminated by providing fake locations. To illustrate, a popular way of attracting customers is to use LBSs to offer discounts, gifts, and free drinks. A customer who visits a cafe or a restaurant usually completes a check-in process using a mobile application to prove that he was at this location in order to get his gift. This, however, can be easily manipulated using a fake GPS software. A location verification scheme was presented and implemented in [6]. Their work focuses on proving location claims in online review platforms, and they claim that adding an additional hardware-based solution to prove location claims is often costly and requires bigger over-head. Hence, they propose a system where users have to take a photo of a restaurant,cafe or a given entity. This photo should overlap with already existing photos which are acquired from previous location claims with respect to right-hand side of the previous images and overlapping should be at least 50%. Previously collected images are stored in a tree-based system in which renewal of the images of a given location is based on asking users to provide new pictures every once in a while. This approach gives rise to certain security problems, however. A malicious user could start to initialize an image tree with a false location of the given entity, as there are no previously collected images on hand. Alternatively, an adversary would collect images that look similar in the image database to overcome this problem of matching the images. Nonetheless, they define their adversary models, and they say given the system if the amount of honest users is more than the dishonest, system will effectively retain location verification quality. It is important to note that this verification approach does not protect the reviewer's location privacy, as it is an online review system. But it shows that an environmental factor,e.g taken image of a cafe, could be used to prove that a user was at a particular location rather than using an additional infrastructure.

In contrast to [6], [36] introduces a collaboration based location proofs between the users by using distance-bounding algorithms which brings up another problem of finding peers in a given environment to create location proofs, as for some systems, it would be hard to find peers in the scarce areas. According to [36], a user who wants to make verification claim for another user creates a location proof by being a witness. In this approach, the terms like prover, witness and probative come into use. Distance-bounding algorithms are susceptible to Terrorist-Fraud attacks, however, they use The Bussard-Bagga protocol proposed in [7] to which they say the most secure distance bounding protocol in existence and not vulnerable to this attack.

3 Designed Solution

In this chapter, we focus on our designed solution by starting off with giving a high-level overview and the description of the protocol. Later, we continue with an elaboration on the receiver and sender sides and show how the protocol operates between two user roles. After the elaboration is shown, we describe our security model in which we justify our security countermeasures and the threat model where we define our potential threats by showing the structural vulnerabilities of our protocol.

3.1 Protocol description and high level overview

3.1.1 High level overview

In this thesis work, we want to assure that a message is geographically restricted while retaining the recipient's location privacy. Let's take a closer look at a simple analogy between Bob and Alice to illustrate the situation where Bob, a security researcher, wants to send a message to Alice at the desired location. The following analogy illustrates a simple use case:

1. Bob wants to send an online love letter to Alice at their favorite cafe.
2. Bob doesn't want Alice to be able to read the message without being at anywhere else than at the cafe he wants.
3. Bob, as a security enthusiast, doesn't want anyone to reveal their location to any other entity, say an application or web service, like other couples do.
4. Bob uses location-based encryption protocol to encrypt the message using the cafe's location and sends it to Alice.
5. Alice gets the message and starts to search the correct location of the message by actively capturing her location at different positions via her phone, etc.
6. If Alice is not at the right place, the cafe in this case, she will not be able to read the message, as it is encrypted with the cafe's location.
7. Alice understands that the message is at their favorite cafe and moves towards the cafe.
8. Alice enters the cafe and decrypts the love letter from Bob.

As can be seen from the above analogy, certain factors play a major role in such a protocol, such as securely transmitting the message between the users, protecting the location privacy, and restricting the information in a specific area. The main question in such a system, however, relies upon how the location information is included in the encryption process, as we want to geographically lock the message. In our work, we aimed at encrypting the message with a symmetric encryption algorithm and using the location information as an entry input in order to derive a symmetric encryption key. Using this approach, we can expect that if a receiver wants to decrypt the message, he should also retain the same symmetric key. To have the same key, he should find the right coordinates which are used to create this encryption key. This approach allows us to define geographical limitations on information using cryptographic building blocks.

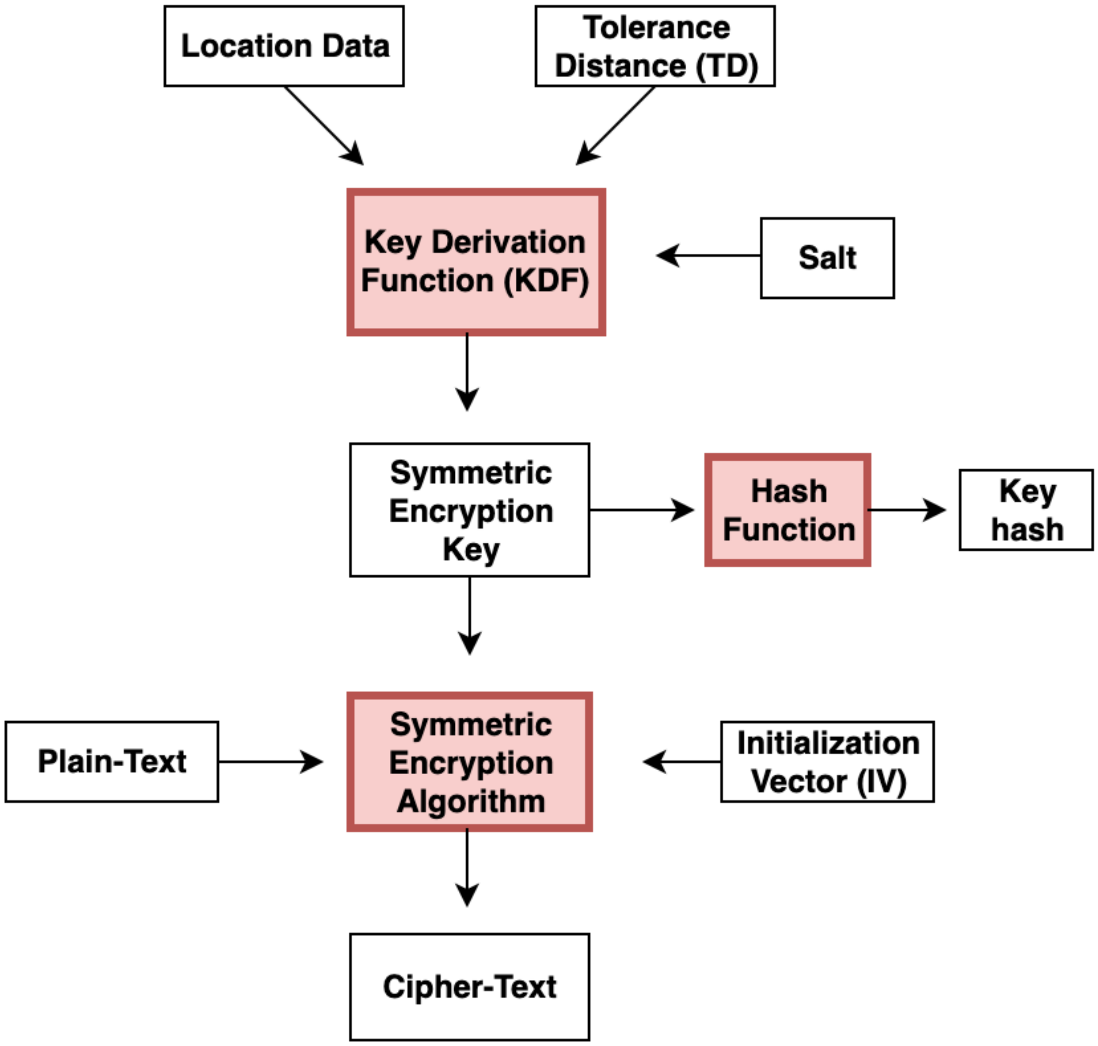

Figure 6: Process of encrypting a given plain-text is presented. Red boxes represent the used cryptographic building blocks to encrypt the plain-text. Location Data, Tolerance Distance(TD) and the Salt are used as inputs to key derivation function. Once the key is derived, a hash function is used to calculate the hash of the key. Later, the derived key is used as an input to symmetric encryption algorithm with the plain-text and initialization vector(IV) to encrypt the plain-text using symmetric encryption algorithm.

It is important to understand that encrypting the message with location data will geographically lock the message in a particular position. Hence, such a protocol should be designed in a way that would allow some tolerance in the decryption range. If this condition is not matched, even a meter of dislocation of the recipient's exact position relative to the message's actual position will yield unsuccessful decryption. Figure 6 shows a high-level overview of the protocol where the symmetric encryption key to encrypt a message is derived from location data.

3.1.2 Protocol description

To have a better and structural understanding, let us break Figure 6 down into individual parts from top to bottom so that we can get an insight into each step as we reason through each component. Earlier, it was said that we use a symmetric encryption algorithm, in which the sender and the receiver agree upon a single (shared) secret (an encryption key); given a message (called plain-text) and the encryption key, encryption produces unintelligible text, which appears to be random (called cipher-text), in order to encrypt our message while still using location data [34]. Now, if we take a step back and take a look at symmetric encryption keys, we see that, essentially, there are two ways to create a cryptographically secure encryption key: with the help of *Cryptographically Secure Pseudorandom Number Generator (CSPNG)* which is a is a Pseudo Random Number Generator (PRNG) with properties that make it suitable for use in cryptography and/or by using a *Key Derivation Function (KDF)* which derives a cryptographically secure key from other secret information, such as another key or an initial secret, like a password. Thus, in order to derive an encryption key, we preferred using a KDF for the scope of this thesis, as we have an initial keying material, which is our location data.

There exists different password-based KDFs (see section 3.3.1 for more elaboration) that take an initial secret, like a password, as a keying material and create an encryption key from that, such as Password Based Key Derivation Function 2 (PBKDF2), Bcrypt, and Scrypt. Bcrypt and Scrypt are very memory exhaustive, by nature designed to require a large and adjustable memory usage to compute efficiently while slowing down the speed rate at which a brute-force attack, in which an attacker tries every value in the possible key space so as to find the encryption key, could operate on. However, PBKDF2 requires less memory usage, which is also important for mobile devices, as it could drain the battery significantly more depending on circumstances[12]. Therefore, we use PBKDF2 as our KDF. As shown in Figure 6, key derivation function takes three inputs as the argument[6]: Location Data, Tolerance Distance(TD) and Salt, respectively. Then it outputs an encryption key of a specified algorithm with a desired key length. Location Data refer to the geographical coordinates of an entity consisting of latitude and longitude values while Tolerance Distance(TD) refers to the distance, in meters, that is used to increase decryption range from the center and allowing some error tolerance. Salt enables reusing the same master key for multiple derived keys with a different Salt per use. Thus, without the knowledge of the Salt, it would be impossible to get the same key when only the initial secret information, the latitude and longitude, is on hand; hence, defeating rainbow-table attacks[7]. In other words, even if the initial key materials, latitude and longitude values, would be the same as inputs inside the key derivation function because the Salt values are unique, this would result in a different encryption key at the end after multiple times hashed inside the key derivation function. As can be understood, what is important is that they should be unique; hence, it boils down to how many different keys we want to derive. Generally speaking, 32 and 64 bits salts are preferred, both of which are unlikely to be exhausted. Therefore, we use 32 bits Salts. Key derivation function is also specified with 1.000.000 iterations in order to slow down this process as much as possible while still providing a good performance. Lastly, we use *Advanced Encryption Standart (AES)* as our symmetric encryption algorithm, as it is still presumed to be

[6]Notice that key derivation functions might have multiple parameters as input, such as output key size, salt, work factor, memory usage and parallelism depending on the selected key derivation function

[7]A deeper insight is given in section 3.3

practically unbreakable[34]. AES is used in counter-mode (CTR) with 128 bits, as it provides less memory usage, meaning less battery usage, for mobile devices, in our case[4, 28].

By deriving a symmetric encryption key using key derivation function as shown in Figure 6, we can now encrypt our messages using a symmetric encryption algorithm. However, let us consider the Bob and Alice analogy again. In this case, when Bob encrypts a message and sends it to Alice. How will Alice understand that the key she has derived from a possible location point is the right key remains as a question, as a wrong key might produce a deceiving output. To solve this problem, we let the sender take a hash of the encryption key and send it to the receiver along with the cipher-text. This way, the receiver will have a reference of the key to compare when he starts to derive keys while searching the correct location. Consequently, as the receiver searches for the correct location, he will create the keys based on different locations and by hashing them, he can now compare it with the key hash he received from the sender. If the key hashes match, he could successfully proceed with decryption. We use Secure Hash Algorithms (SHA)-256 to get a hash of the key as well as for the key derivation function, as SHA-1 is broken[24]. In the last step, the message is encrypted using a symmetric encryption algorithm. As it is illustrated in Figure 6, it takes three inputs which are plain-text, encryption key and Initialization Vector (IV). Plain-text refers to the message to be encrypted where the encryption key is the symmetric key to be used in encryption, and the initialization vector is used to prevent repetition in data encryption, mitigating dictionary attacks to find patterns and break the cipher-text. Size of 16-bytes IV is used, as it is mostly parallel to the block size of the given encryption algorithm.

3.1.3 Location data in key derivation process

Since we have an insight into how does the protocol work, we can now delve into the process of constructing the encryption key from location and tolerance distance data. Figure 7 illustrates an example of how a symmetric encryption key is derived from latitude and longitude values along with tolerance distance. First of all, let us start by examining the location data. As mentioned before, the receiver actively captures his location using his mobile device. A typical representation of location data is as follows:

> **N 4741.3501'** could be read as *47 degrees 41.3501 minutes North*
> **E 911.3182'** could be read as *9 degrees 11.3182 minutes East*

Above example is called *Degrees Decimal Minutes*[8] representation of location data. This representation allows latitude values to be between 0 to 89 degrees while allowing longitude values to be in the range of 0-179 degrees. Hence, when we read them, we should start by taking the point sign as a reference. As minutes can be maximum of 59, the first two digits located left-hand side of the point sign can be used to show minutes along with floating-point numbers. The other decimal numbers, in the beginning, shows the degrees part of the location data.

In the second step, we should include our *Tolerance Distance(TD)*, in meters, to the location data. After multiplying the latitude and longitude values by 10.000, we divide the resultant with the desired tolerance distance. A 1 meter change corresponding to latitude and longitude values are 5,4 and 6[9], respectively[25]. By doing so, we are creating quadrants based on our tolerance distance. In other words, we allow multiple location points to fall under the same group based on tolerance distance so that we can allow them to decrypt information if they are in the same quadrant. This way, we can now decrypt the message in a given range rather than having to decrypt the message at a very specific position. We can now also touch on why we multiply the location data with 10.000 in the beginning. The reason is related to the accuracy of the position. As we see in Figure 7, latitude and longitude values can have up to 4 digits after the point sign. Hence, when divided by tolerance distance, if it is multiplied by other value than 10.000, let us say 1000, the resultant will

[8]Essentially there exists two more possible representation of the location data: Decimal Degrees(DD) and Degrees Minutes Seconds(DMS). See Appendix A for more insight

[9]See Appendix B for detailed explanation.

be a lot less precise than when it could be multiplied by 10.000, as we lose rest of the digits, making it less accurate. This would mean that we would refer to more rough addressing of the given location point.

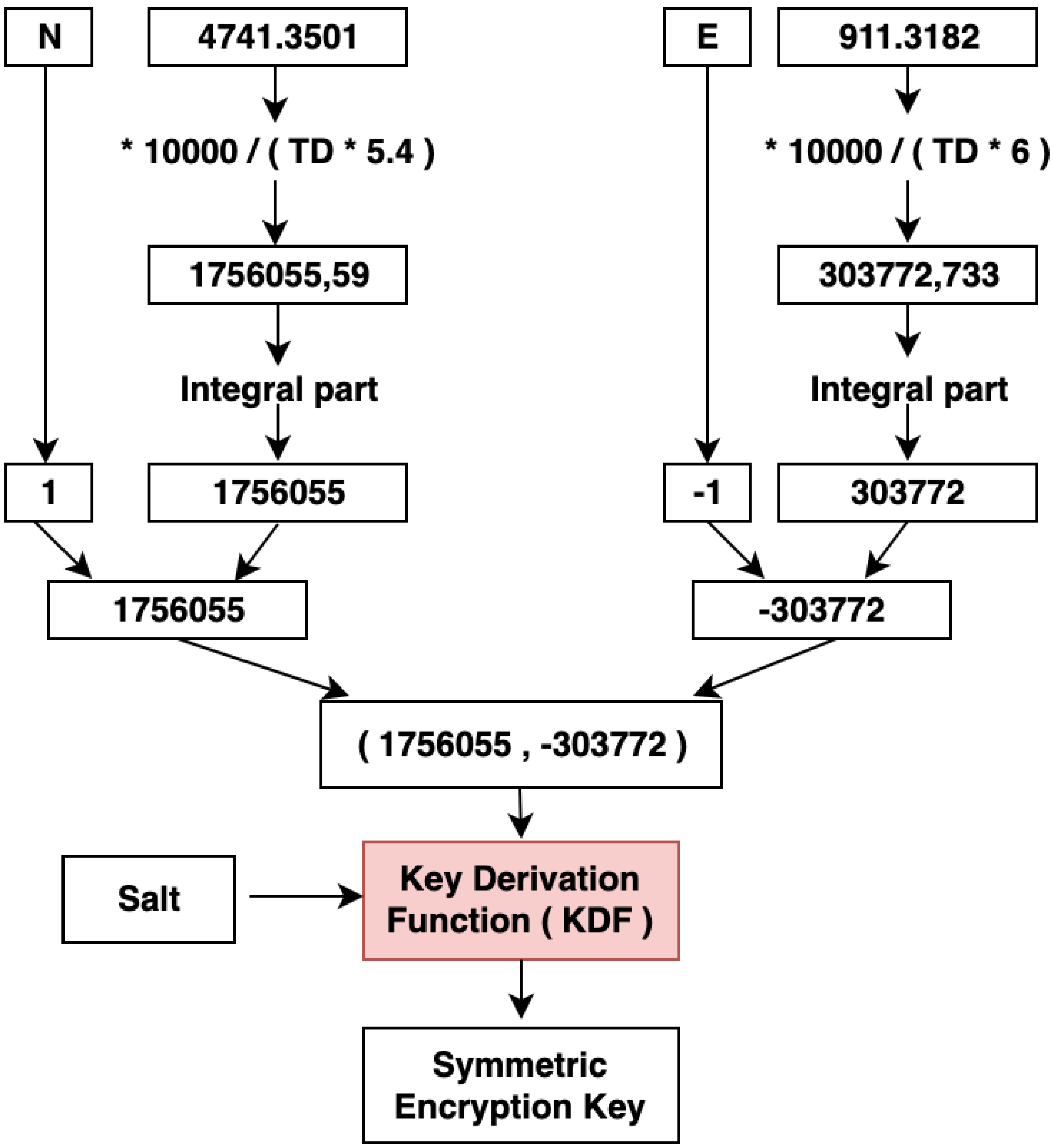

Figure 7: Latitude and longitude data are combined with Tolerance Distance(TD) to create a quadrant which allows other location points within the tolerance distance to create the same input for the key derivation function. Later, this input is used with the Salt for the key derivation function which creates the symmetric encryption key. In this example, tolerance distance is assumed to be 5 meters and corresponding inputs are calculated.

Let us give an example for this purpose and consider Figure 7 for the calculations. Assume a reference point of N 4741.3514. Let us also assume that two location points A and B with 4 and 15 meters relative distance to reference point[10] having location values of N 4741.3536 and N 4741.3595, respectively, with a tolerance distance of 5 meters. We can now examine what happens if we multiply the location data with 1000 instead of 10000.
Subsequently, we need to apply the steps shown in Figure 7 both for A and B locations.

1. Location A: 4741.3536 * 1000 = 4741353.6
2. Location A: 4741353.6 / (5 * 5.4) = 175605.68
3. Location A: Taking the integral part: **175605**
4. Location B: 4741.3595 * 1000 = 4741359,5
5. Location B: 4741359,5 / (5 * 5.4) = 175605,907
6. Location B: Taking the integral part: **175605**

Above example shows that two location points with 4 and 15 meters of relative distance to reference point result in the same resultant integer value although the tolerance distance is 5 meters. Hence, the left side of the input of the key derivation function, shown in Figure 7, would be the same for both point A and B although point B is outside of the 5 meters tolerance distance range while A satisfies this condition. If, however, we do the same steps with multiplying both location data by 10.000, corresponding values will be 1756056 and 1756059 for A and B, respectively, which clearly shows that in this case result is more precise from location accuracy stand point. Thereby, the input result has a higher preciseness and more accuracy instead of rounding off the numbers by lessening the sensitivity.

Having included Tolerance Distance(TD) and taken the integer part of our location data, we can now also add the location sign. Typically, location signs are North(N) and South(S) for latitude values while West(W) and East(E) for longitude values. Location sign is set to be 1 if located on Western and/or Northern hemispheres; whereas, it is set as -1 for Eastern and/or Southern hemispheres in order to be added into location data. Since we have the resultant location data changed based on tolerance distance, we can now multiply the resultant with location sign in order to add hemisphere information to this data.

Once all calculations are made, both of the latitude and longitude values are used as inputs to key derivation along with salt so that the symmetric key is generated.

3.2 Elaboration on the sender and receiver sides

In the previous chapter, we looked at the general overview of the protocol in addition to the process of constructing a symmetric key using location data. How does the protocol work between the sender and the receiver is presented in this chapter.

Figure 8 presents an overview of the communication between a receiver and sender. Firstly, as mentioned before, the sender decides two values: location data which is whereabouts of the message to be encrypted and tolerance distance in meters specifying the decryption range. These inputs are used to feed the key derivation function, along with the Salt. Once the key is derived, the sender uses the hash function to get a key-hash of the derived key so that the receiver can have a reference when he tries to decrypt the message by constructing his own key. After, an initialization vector(IV) is used with encryption key as an input to the symmetric encryption algorithm. Plain-text is, thereafter, encrypted and sent through the end-to-end encrypted channel.

[10] Distances are calculated using Haversine Distance Formula, see Appendix B, and assumed that they are located on the same longitude

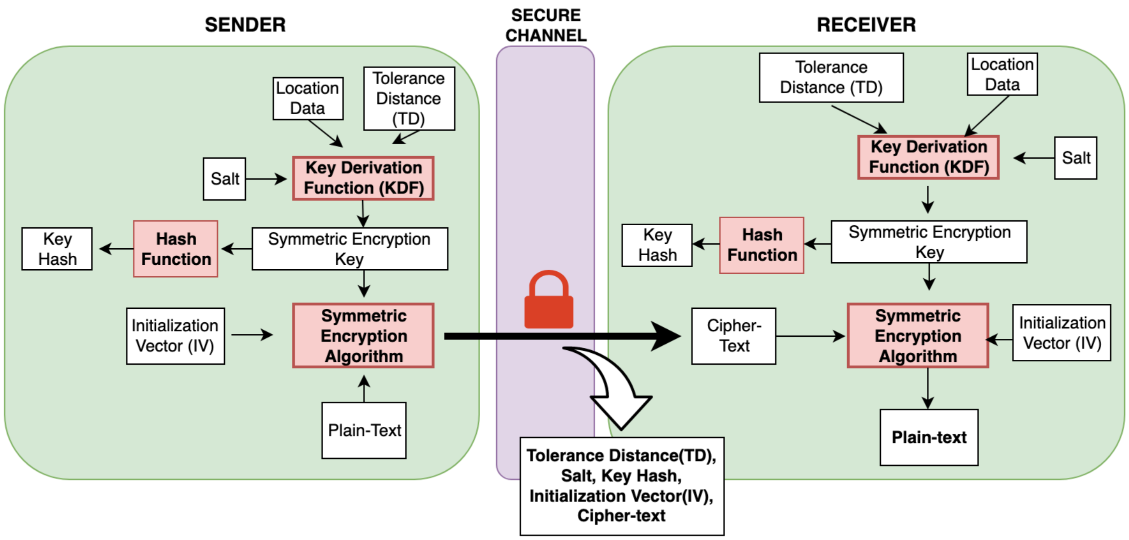

Figure 8: In order to derive the same encryption key, Tolerance Distance(TD), Salt, Key hash, Initialization vector(IV) and cipher-text is sent to the receiver using End-to-end Encryption (E2EE). Using these information, the receiver will have everything he needs to derive the same key. Importantly, he can use the key hash to check if he created the correct decryption key.

Logically, the receiver should possess the same values to derive the same key. This includes the Salt and Tolerance Distance(TD) for the key derivation function. Later, he should have a reference to compare his derived key with the actual key's hash which he should get from the sender. Lastly, as he has the key and the cipher-text, he should also possess the same initialization vector(IV) to decrypt the message. Summarily, this implies that receiver should get *Tolerance Distance(TD), salt, key hash, initialization vector* along with the *cipher-text* to generate the same key. Hence, the sender sends all of this information through this secure channel to the receiver, as they are very crucial to create the same cryptographic key.

3.3 Security model

In this chapter, we describe our security countermeasures needed to ensure our security requirements.

3.3.1 Computational expensiveness

Since a receiver can also act as a malicious user, we should investigate how we use location data in the key derivation function. Typically, the receiver might want to decrypt the message without being at a particular location. In the current version of the protocol, location data are used as inputs to key derivation function; however, verification of the location claim of an entity is not checked. Hence, they can spoof the protocol by providing a lot of fake location through different approaches, such as using a fake Global Positioning System (GPS) or simply by brute-force approach. Now, as we have shown before, we use location information as an input to key derivation function. Inherently, cryptographic key derivation functions have multiple purposes based on their usages:

- **Key Separation:** An essential usage for key derivation functions. Here the key derivation function is used to derive multiple keys based on another cryptographically secure encryption key. There might be several use cases behind this, such as an authentication server communicating with multiple clients might need unique keys for each communication. Also, for a certain security reason, a server might need to handle each communication by a new encryption key. When used together with the Salt, unique per session, key separation allows us to derive multiple keys.

- **Key Expansion:** Key derivation functions can also be used to change the current configuration of the given key into a specified format. In specific cases, 128-bit encryption key would be converted to be a 256 bit key under the assumption that 256 bit provide more security under certain conditions.
- **Key stretching:** This one is the specific usage of key derivation functions where an initial passphrase, say a user password with low entropy like 30 bits, is intended to be used for the encryption and the decryption while still making it hard to break encryption by guessing the passphrase by brute-forcing.

In particular, as shown above, key stretching is used for all password-based key derivation functions to derive an encryption key from an initial keying material. By making this process purposefully slow using a very high iteration number, we can slow down brute-force attacks, in which an attacker tries every possible input in the existing key space. As we would have computationally expensive key derivation function, it would slow down the key generation time millions time more compared to a key derivation function with a really small iteration number. Therefore, we use a computationally expensive key derivation function to deter a receiver from flooding fake location values.

3.3.2 Nonce information

In the previous section, we showed different key derivation functions; in particular, we touched on how we take advantage of key-stretching in key derivation functions. As we mentioned before, key derivation functions have multiple parameters that act as auxiliary information to further increase the security and the capabilities of the key derivation functions. In our case, we take advantage of using Salt to prevent a receiver from performing rainbow-table attacks in which the attacker has a precomputed table of vast amounts of strings that could be used as inputs to key derivation function. Afterward, attempts can be made to compare the outputs of the hash function to find a match so that he can look backward and find the corresponding input. In section 3.4.2, we show the underlying problem with the location data. It is shown that location data are very limited in terms of key space due to geographical limitations of the Earth and the tendency of the human-being on urbanizing in particular areas. Hence, these small set of values are very susceptible to be guessed. Accordingly, when a malicious user who would want to decrypt the cipher-text could follow two tactics. He would either try every possible input inside the key derivation function and try to decrypt the cipher-text, which is called the brute-force attack, or he could try a rainbow-table attack. The first case is very costly; thus, the second attack type would be very easy for an attacker to perform. Under the circumstances, we use the Salt to discard these pre-calculated tables if an attacker tries to guess the limited location data. As the Salt is not known to him before, he would have to calculate a new table for each salt, which makes it as expensive as a brute-force attack. Hence, using Salt as an input to the key derivation function, we mitigate the chance of rainbow-table attacks.

Similarly, when a plain-text is encrypted with AES, depending on the mode of the AES, the cipher-text might be susceptible to certain attacks and/or reveal information. For example, in EBC(Electronic Code Book) mode the plain-text and the encryption key always produces the same cipher-text; however, we use IV(Initialization Vector) in CTR mode and that helps us to create different cipher-texts even if the same message is encrypted with the same key over and over again, different IV per usage. Likewise, if the attacker doesn't have this information, he wouldn't perform such an attack on the data.

3.3.3 Secure communication channel

To mitigate any unauthorised dissemination of the data from an outsider, see section 4.2.1 for details on the insider and outsider attackers, we use E2EE in our protocol. Hence, we facilitate a secure channel allowing communication between a sender and receiver. This way, any data that are transmitted between the sender and receiver will be only readable in these ends. To accomplish such a task, a message is encrypted using an asymmetric-key encryption, also known as public-key cryptography algorithm. In E2EE, there exists two mathematically related keys: public and private key. Each user keeps their private key to themselves and lets other users know their public key. A message is encrypted by a sender using the receiver's public key. The only way to decrypt this message will be to use the corresponding private key held by the receiver. An overview of the development of E2EE systems has been well-presented in [11]. In this sense, any intermediaries will be prevented, as they do not have the corresponding private key. Therefore, we use this type of protection from outsiders to eliminate their chances of tampering as mentioned in the previous section where adversaries are defined.

Location data lacks in entropy[11], and as we've shown in the previous chapters, we use this data with the Salt to prevent rainbow-table attacks. If the Salt is obtained by an outsider attacker, he would also easily perform a rainbow-table attack, and he would have a significantly reduced key space to search due to small key space of the location data. Besides, obtaining other components like key hash and the cipher-text would allow the attacker to deduce some information under certain conditions as well as operating attacks like cipher-text-only attacks. Considering all these, using E2EE helps us to mitigate these problems.

3.4 Threat model

In this chapter, we will look at the threat model in which we show the structural vulnerabilities of our location-based encryption protocol.

3.4.1 Adversaries

In this section, we will define two adversaries and mention what they are capable of with regards to manipulating the protocol using possible flaws. The first adversary could be an insider user who receives a message from a sender. In this case, the receiver could potentially try to read messages without fulfilling certain requirements, such as being at a particular location decided by a sender to decrypt an encrypted-text. He can do this by using fake GPS software or any other solution which might manipulate the device and show its location as if it was in a different area. This way, a malicious user would try to decrypt the messages by providing these fake locations.

Alternatively, an outsider who would want to learn the content of the message would try to tamper with the information. As the communication channel is secured using E2EE, he simply can't decrypt the cipher-text because he doesn't possess the corresponding private key. However, he can intercept the conversation and act as a receiver rather than trying to break the encryption algorithm. This way, he can encrypt the incoming messages from a sender by using his public key and then resend them to the actual receiver using the receiver's public key. If he doesn't get caught due to delays caused during this process, he could make the sender and receiver believe that they are directly talking to each other.

3.4.2 Key derivation function input space

We use latitude and longitude data to specify the geographical location of an entity. The data, then, are used as an input to key derivation function along with the Salt. Given that salt is already possessed by the receiver

[11] see section 3.4.2 for the detailed reason

at the time of decryption, the receiver could significantly reduce the searching area of a place to be decrypted by trying relatively small location data due to world's geographical facts, such as the fact that roughly only 3% of the whole Earth's surface is occupied and urbanized by humankind[2]. Accordingly, this constitutes a direct threat against the key space of the protocol, as the Salt is already known by the receiver. Let us calculate possibility of finding the right location with respect to different tolerance distances of 10,20,50 and 100 meters, based on the Earth's surface area, $5,1 \times 10^8\ km^2$, i.e., $5,1 \times 10^{14}\ m^2$ and the fact that only 3% of the Earth's surface is urbanized. We can calculate the total number of different possibilities by dividing the area of The Earth's urbanized area by the area of a circle specified by the tolerance distance.

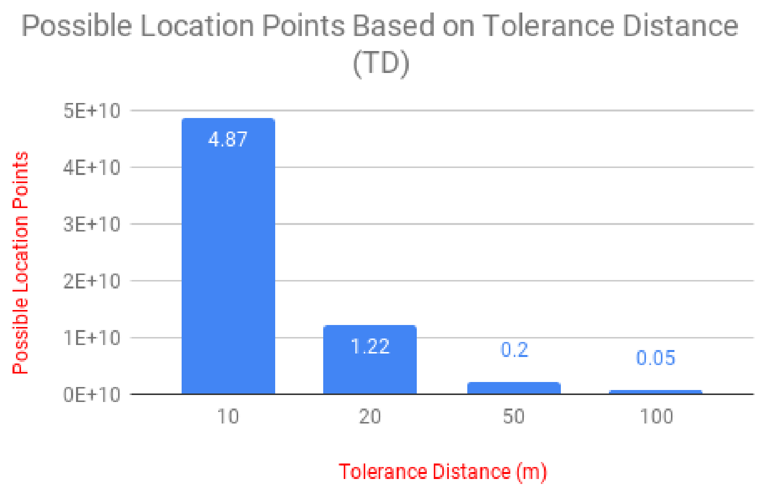

Figure 9: Possible number of locations based on tolerance distance is calculated depending on the Earth's surface area and the fact that only 3% percent of the surface is urbanized. It's shown that due to the fact that Earth is mostly covered with water sources, number of possible locations significantly reduced, when the tolerance distance is increased.

When the tolerance distance is 10 meters,

$$\frac{1}{5,1 \times 10^{14} m^2 \times 0.03/(3.14159 \times 10^2) m^2} = \frac{1}{4.87 \times 10^{10}} \tag{1}$$

For 20 meters tolerance distance,

$$\frac{1}{5,1 \times 10^{14} m^2 \times 0.03/(3.14159 \times 20^2) m^2} = \frac{1}{1.22 \times 10^{10}} \tag{2}$$

For 50 meters tolerance distance,

$$\frac{1}{5,1 \times 10^{14} m^2 \times 0.03/(3.14159 \times 50^2) m^2} = \frac{1}{0.2 \times 10^{10}} \tag{3}$$

For 100 meters tolerance distance,

$$\frac{1}{5,1 \times 10^{14} m^2 \times 0.03/(3.14159 \times 100^2) m^2} = \frac{1}{0.05 \times 10^{10}} \tag{4}$$

As can be seen from above four equations and Figure 9, based on different tolerance distances, the bigger the tolerance distance, the smaller the key space is. The probability of breaking such a key would be trivial for an insider attacker if location information would directly be used in a regular hash function with current processors' speed. Therefore, we use a computationally expensive key derivation function to dissuade a receiver from brute-forcing this relatively small key space[12].

3.4.3 Secrecy of communication channel

In section 3.3.1, usage of the Salt was explained. We saw that if the Salt, unique per message, is obtained, the possible key space for the key derivation function is significantly reduced. Therefore, if the Salt were known to an outsider attacker, the same brute-forcing tactic would also apply to this type of malicious user. Also, the other transmitted information, such as IV, cipher-text and key hash are very important. Hence, the whole secrecy of the salt, including the other auxiliary data, relies on how securely E2EE is constructed, as knowing salt would drastically reduce the possible input space for the key derivation function. Theoretically, E2EE promises this by disabling any intermediaries from being able to eavesdrop and/or tampering with the message. This way, 16-byte salt value is not known to an outsider, and rainbow attacks are eliminated, as it would be a very expensive calculation for the attacker with regards to the computational cost of an attack. However, a recent study claims that even if E2EE is promised by a service, depending on the implementation medium, say a mobile application or a desktop application, compromising might still be subject to interpretation[23]. However, we leave this topic for another research study, for the scope of this master thesis.

3.4.4 False positive and false negative values

As one can expect, any attempt made during the decryption should be successful only if the attempt has been made within the tolerance distance. In Figure 7, we see that when the encryption key is constructed, the resultant integers from latitude and longitude values represent a quadrant in which the encryption key stands. These quadrants can be thought of as individual parts of a grid which represents the Earth's surface. Based on our observations, when the encryption key is constructed on the receiver's side, the same quadrant creates the same key logic fails as we get closer to the boundaries of the quadrant. Accordingly, we observe that resultant integers which are very close to boundaries of a given quadrant fail to build the same encryption key, causing to have false-negative values and unsuccessful decryptions.

To illustrate, as shown in Figure 7, the integer part of the fractional number is taken after latitude and longitude values are divided by the tolerance distance. However, we see that depending on the location data provided by the receiver during a decryption attempt, this resultant integer might have a difference with a maximum range of 0 to 1 compared to the value that is calculated by the sender although the receiver's location is within the tolerance distance.

As an example, let us consider a location of 4741.3503 N latitude with a tolerance distance of 10 meters. While the resultant integer for this position would be 878027, another location which is ten meters away from this reference point with a latitude value of 4741.3555 N would have 878028 as the resultant integer. Therefore, even though second location assures that the location is within the tolerance distance, due to false-negative value situation it wouldn't decrypt the cipher-text successfully. In order to mitigate this problem, we allow a receiver to also check the left and right adjacent quadrants as well as its own quadrant. This solves this problem, however, it also allows some of the receivers to decrypt the information even if they are not within the tolerance distance, causing false-positive values. Considering the above example with 4741.3555 N position, the resultant integer yields as 878028 when multiplied by 10.000 and divided by the tolerance distance. To include the adjacent quadrants, 878027 and 878029 resultant integers would also be

[12]For implementation purposes, key derivation functions could further be made computationally expensive by increasing the memory usage or iteration number

tried as an input to the key derivation function. If the same process is applied to both latitude and longitude values, total of 9 different combinations should be provided to key derivation function as an input.

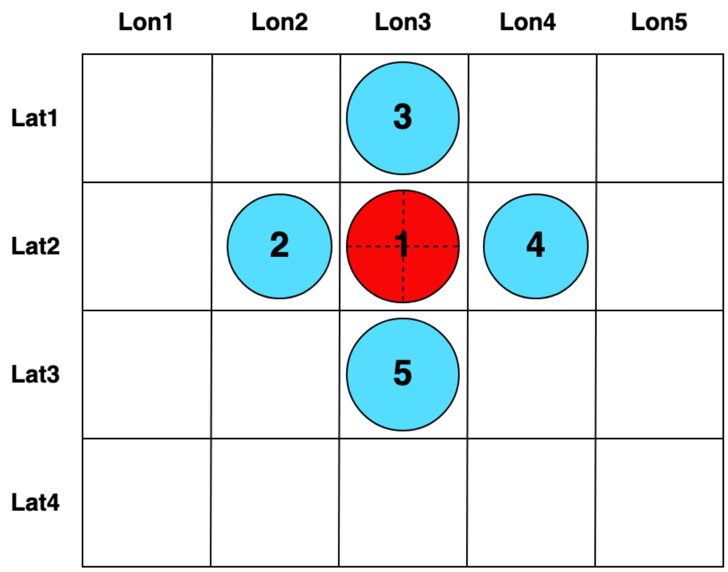

Figure 10: In the given grid, red circle (1) shows the quadrant in which the encryption-key stands. Even though Tolerance Distance (TD) is used to increase the decryption area, the calculated quadrant doesn't cover the whole area inside the quadrant. As it gets closer to boundaries of the current quadrant, the location points which are close to the boundaries fail to fall in this tolerated area. Hence, when the receiver calculates his own key, he should also check left (2) and right (4) quadrants for the latitude, as well as the north (3) and south (5) quadrants for the longitude. This way, the receiver should try 9 different combinations to include adjacent quadrants both for latitude and longitude values.

In Figure 10, an example case is illustrated. A sender is located at a certain position and key is built based on this position and the tolerance distance. As explained earlier in the mathematical example, the sender's quadrant fails to cover all of its values as we get closer to boundaries. Accordingly, other values which are still inside the same quadrant but very close to boundaries wont be able to build the same key, which will cause false-negative values. Alternatively, as the receiver will be able to check total of 9 quadrants, in some cases he will be able to decrypt the encrypted-text although he is not in the right quadrant, causing false-positive situations.

3.4.5 Verification of location claims

One of the biggest problems with Location Based Systems (LBSs) and location-dependent encryption systems are how the location verifications are made against the location claims. Because in reality, cryptographic components often provide very high security by successfully encrypting the plain-texts and matching the security requirements; however, the end systems are weak to manipulation, such as stealing the encryption key from a user's phone. In general, many of the location-dependent systems have a huge problem with the verification of the location claims. This is due to the fact that it is very easy for mobile devices to use manipulation tricks to change their current location. Hence, a device would act as if it was in a different location. This creates a huge problem in the case where location claims aren't securely verified. Depending on a system, there might be several solutions to prevent this on the application level. However, in the encryption level, we do only encrypt the information based on the location data. This means the verification of a location claim is not checked; therefore, very susceptible to manipulation. Current solutions often include cooperation with nearby peers to create so-called "location-proofs" and/or witnessing approach, which uses an additional server where location verifications are made, such as using triangulation. However, we stress that location privacy is our top priority and shouldn't be revealed to the other entities apart from the original recipients. In summary, our protocol lacks in location verification, and we leave this as a further study problem for another study.

4 Evaluation

In this chapter, we first give an insight into performance analysis of the protocol with respect certain parameters, such as time, estimated data consumption and the auxiliary parameters affecting the encryption and the decryption. Then, we conclude it with a security analysis. Importantly, for implementation and research purposes, we made LOCKUM publicly accessible[13].

4.1 Performance analysis

In this section, we will focus on performance evaluation. We start by examining the duration for several cryptographic operations in the protocol. Earlier, we mentioned that key derivation function has auxiliary parameters depending on which computational expensiveness changes. We also know that number of iterations in the key derivation function has a crucial impact on the key derivation time, which directly influences the speed rate at which the brute-force attacks operate on. As small iteration numbers would lead to quicker key derivation, an attacker will have the opportunity to do faster calculations. According to NIST(National Institute of Standards and Technology) publication, maximum iteration number should be 10,000,000, where user-perceived performance is completely ignored while the minimum number should be as little as 1000[35]. However, this publication was written in 2010 and considering the fact that computation technology shows a significant increase due to recent developments in hardware industry, we consider 1000 iterations as a highly outdated number. Hence, we will test iteration numbers of 10.000, 100.000, 200.000, 500.000 and 1.000.000, as they seem to be used in various systems. We used 2017 MacBook-Pro(2,3 GHz Intel Core i5 with 16 GB 2133 MHz LPDDR3 Memory) as our testing device to encrypt a 30 byte plain-text: "this is a really awesome test." with different desktop browsers, such as Google Chrome(Version 75.0.3770.80), Mozilla Firefox(Version 67.0), Internet Explorer(Version 11), Microsoft Edge(Version 44.18362.1.0) and Safari(Version 12.1.1)[14].

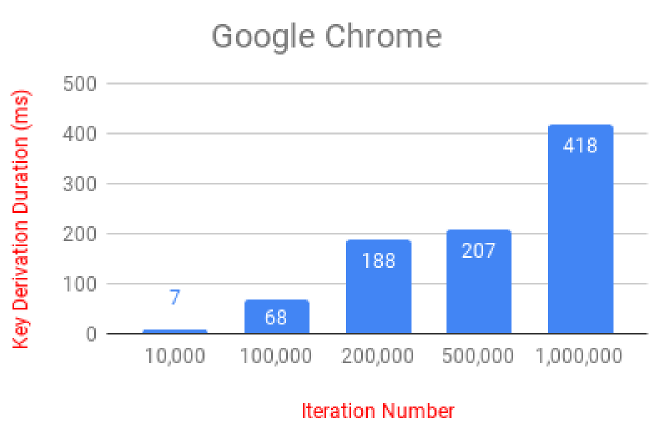

Figure 11: Key derivation time based on different iteration numbers for Google Chrome Browser

[13] Available at https://github.com/disy/lockum

[14] Internet Explorer and Microsoft Edge browsers are tested using Safari browser's developer-simulation tool, as Internet Explorer and Microsoft Edge browsers are not launched for Mac-OS which is what we use as our testing device.

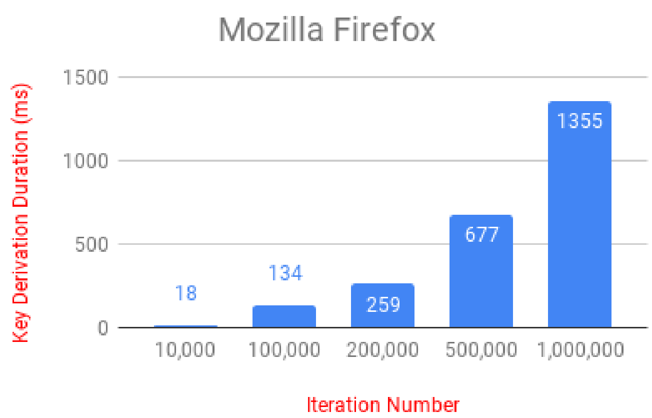

Figure 12: Key derivation time based on different iteration numbers for Mozilla Firefox Browser

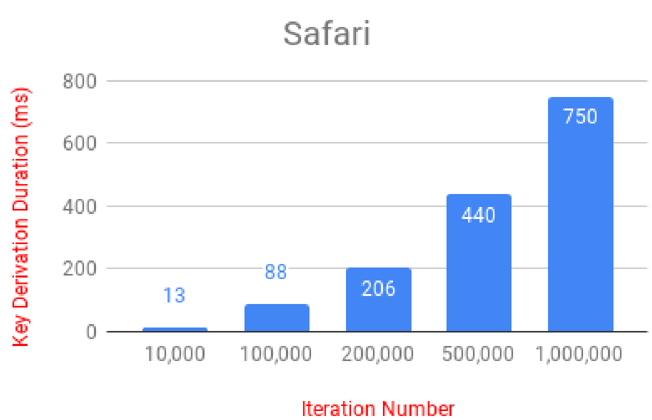

Figure 13: Key derivation time based on different iteration numbers for Safari Browser

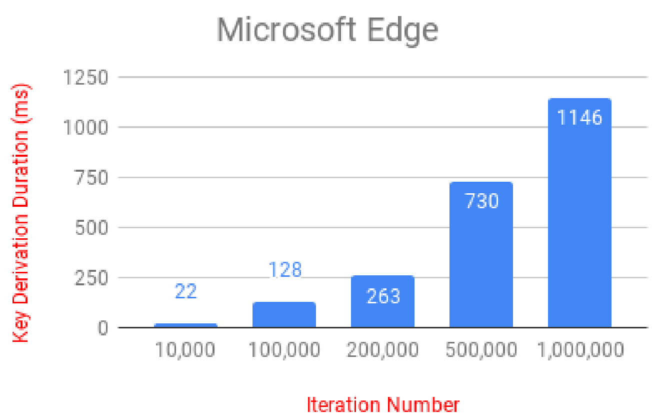

Figure 14: Key derivation time based on different iteration numbers for Microsoft Edge Browser

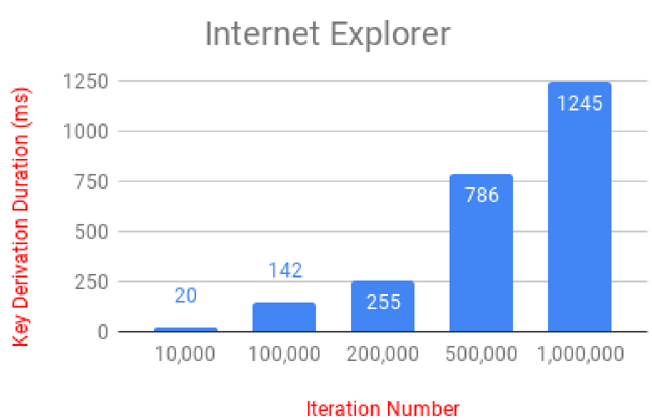

Figure 15: Key derivation time based on different iteration numbers for Internet Explorer Browser

In Figure 11,12,13,14 and 15 we show different durations based on different iteration numbers needed to derive a key. Google Chrome shows a significant superiority with respect to the key derivation duration compared to the other browsers. Alternatively, to see performance differences, we used Advanced Encryption Standart (AES) encryption with different modes, such as GCM(Galois/Counter Mode), CBC(Cipher Block Chaining Mode) and CTR(Counter Mode) and made multiple tests on these browsers. However, the difference between the different AES modes was often fluctuating and insignificant; thus, we couldn't find any correlation.

Another important duration test is how the protocol operates on different platforms during different cryptographic operations, such as key derivation, encryption and decryption, as it is intended for multiple platforms. In order to test this, we used three platforms to test a 30 byte plain-text "this is a really awesome test." on Google Chrome's browser:

1. **MacBook Pro(2017):** 2,3 GHz Intel Core i5 with 16 GB 2133 MHz LPDDR3 memory, MacOS Mojave 10.14.5 Operating System

2. **Samsung Galaxy Tab S2:** 1.3GHz octa-core with 3 GB memory, Android 7.0(Nougat) Operating System

3. **Iphone 7:** Apple A10 Fusion chipset with 2 GB memory, IOS 12.3.1 Operating System.

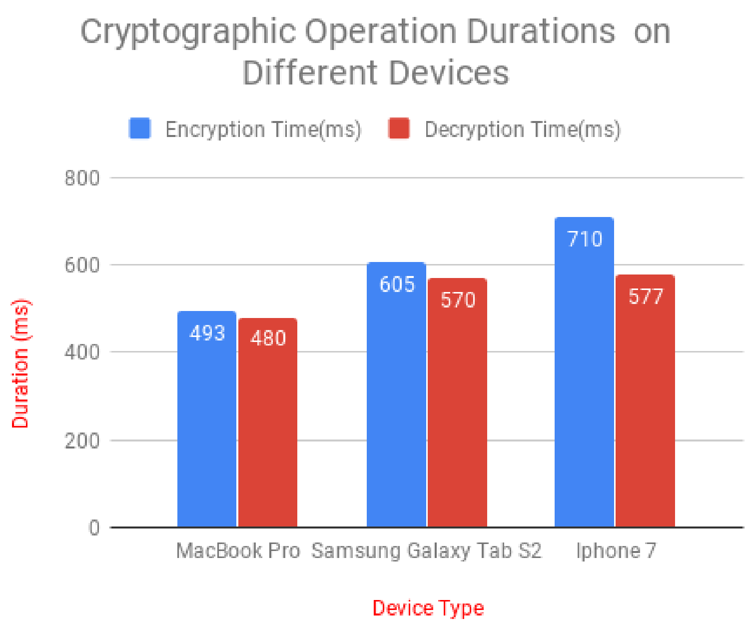

Figure 16: Encryption and decryption operations, in fact, build upon combination of key derivation and encryption/decryption operations. Hence, only the encryption or decryption operation itself takes as little as 1ms, and key derivation duration accounts for the most of the duration during an encryption or a decryption process, as the keys are first derived and then used.

In Figure 16, performance differences are presented among different devices. MacBook-Pro is faster than Iphone 7 with more than 200ms difference; however, it should be noted that as Google Chrome Browser has different releases for each platform, such as Android, IOS, and MacOS operating systems, the difference might not be fair and is subtle between the platforms, as it can be seen from the figure. This is most likely related to different JavaScript engines used in different platforms. Even so, the duration shows an increasing tendency, as the laptop is a powerful machine compared to other two devices, resulting more than 100ms quickness while mobile devices having longer decryption and encryption times.

Moreover, all of our test cases were measured based on the assumption that the receiver and sender are in the same quadrant; thus, during the key constructing process, only the main quadrants are checked. However, as described in section 3.4.4, depending on the quadrant the receiver in, up to 9 keys could be constructed in order to prevent the false-negative values to allow a successful decryption. If a receiver creates 9 keys and tries to decrypt a cipher-text, the decryption time would take up to 9 times as much compared to a perfect scenario in which the cipher-text and the receiver are in the same quadrant. To test this, we used 2017 MacBook-Pro(2,3 GHz Intel Core i5 with 16 GB 2133 MHz LPDDR3 Memory) as our testing device to test a 30 byte plain-text "this is a really awesome test." on Google Chrome's browser comparing different quadrants. In Figure 17, it is shown how it takes more time for adjacent quadrants to create the right decryption key. As can be seen from the figure, for the far end quadrant it takes almost 3700 milliseconds which is a very long duration. Notice that this performance is due to the fact that we use a very high iteration number in our key derivation function, and the actual performance might be even worse on the other browsers from time stand point. Further optimizations could be made depending on the iteration number.

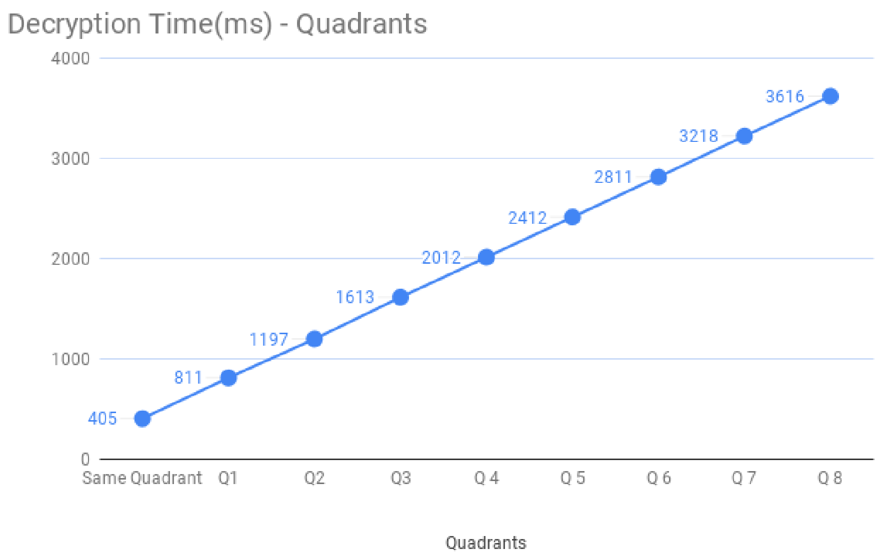

Figure 17: In this figure, decryption time is tested against different quadrants with respect to cipher-text's actual location using Google Chrome's browser. Quadrants are the neighbour cell's calculated as mentioned in section 3.4.4. As it is seen in the figure, the further quadrant it is, the longer it takes to calculate corresponding decryption key.

We should also investigate the usage of such a protocol in real world applications from data consumption perspective, as data consumption is also an important factor. Let us consider a scenario, in which a shopping company distributes coupons to local people in a city with 1.000.000 inhabitants. Let us assume that the company has a weekly discount coupon distribution policy for about 250.000 people that are already saved in their user database. Let us also imagine that a discount coupon consisted of an encrypted coupon code in addition to an advertisement message which corresponds to total of 750 characters. In [37], it is shown that a typical message with Latin Alphabet consisting of 50 characters accounts for some 425 byte data consumption in popular instant messaging application case, WeChat[15]. Also, each additional 16 characters adds up 16 bytes more to the total consumption. Using aforementioned assumptions, we can define the weekly data consumption of a company:

A weekly data consumption = (number of requests) x (data consumption)

We also know that:

Number of the requests: 250000

[15] https://www.wechat.com/en/

An advertisement message consisting of 750 characters: 12226 bytes.
A weekly data consumption = 250000 x 12226 = 3056.5 MB

As can be seen from the above equation, a weekly data consumption for a company to send discount coupons for 250.000 people would cost more or less 3 GB of data, which is very affordable in comparison to traditional ways, such as handing out the discount coupons using a regular piece of paper within the company.

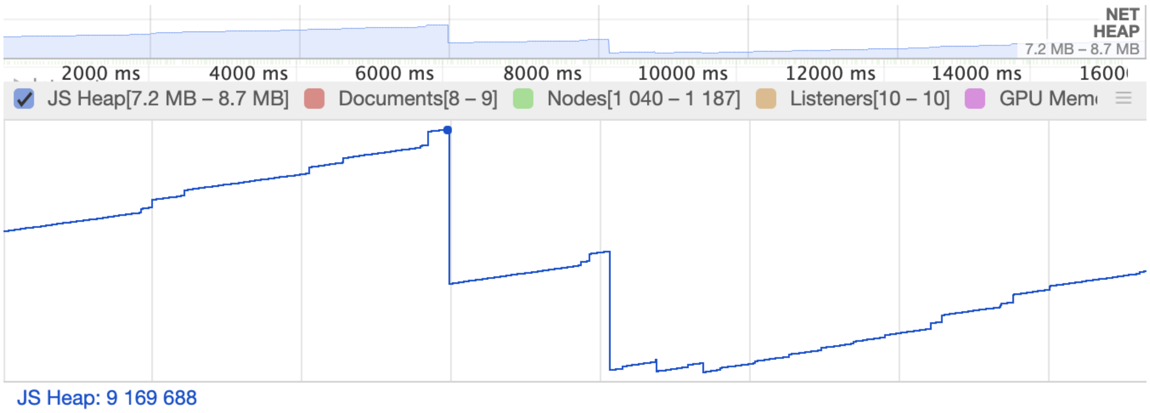

Figure 18: Figure shows the memory allocation caused by JavaScript file on Google Chrome Browser. During an encryption and decryption operation where the encryption is done within the browser, memory allocation is fairly low and battery efficient as the demo page uses at most 7-8 MB of heap during the test.

Lastly, building cryptographic building blocks while providing functionality to mobile devices should be memory efficient, as more memory usage would mean a faster battery consumption for mobile devices. For the scope of this master thesis, we implemented LOCKUM using Web Crypto API [16] which allows developers to use cryptographic operations inside the browser using JavaScript and made it publicly available for research purposes [17]. In Figure 18, memory heap allocation of our implementation is presented during our test with our test device 2017 MacBook-Pro(2,3 GHz Intel Core i5 with 16 GB 2133 MHz LPDDR3 Memory). JavaScript heap consumption shows up to 9 MB usage during cryptographic operations, such as encryption of a given text and decryption of the cipher-text. As can be seen from the figure, memory allocation increases in the beginning; however, after the encryption operation succeeds, the garbage collector frees the memory. Thereafter, decryption operation takes place which triggers the memory allocation again. In summary, we see that cryptographic operations are not a big burden for devices with respect to memory usage.

4.2 Security analysis

In this section, we will provide a thorough security analysis of our location-based encryption protocol based on our security policies which we defined as six conditions in section 1.2 for a secure location-based encryption protocol.

[16] https://www.w3.org/TR/WebCryptoAPI/
[17] https://github.com/disy/lockum

4.2.1 Adversaries

Primarily, we have two possible target groups whose malicious purposes should be thwarted. As can be seen from Figure 8, once the sender sends the data, there are two possible cases where information can be tampered with. In the first case, theoretically, when the data are transmitted, an outsider might purposefully intercept the conversation with the intention of eavesdropping and even stopping the transmission of the ongoing packets. In the second case, however, a receiver would act as a malicious user in order to read messages without being at a particular location, as needed. He could do this by providing different locations without being at those locations points with the help of fake Global Positioning System (GPS). Furthermore, depending on the area they reside, say a small town, he could detect the possible locations where message could be encrypted, and for the second and the third times, he could use these obvious locations to decrypt the message without being at there. Hence, we used these countermeasures to prevent an outsider from tampering with the transmitted data and to prevent an insider attacker from spoofing multiple location points in order to read the message.

4.2.2 Location authentication

In this section, we analyse the protocol's security from location authentication viewpoint. In section 1.2, we defined certain conditions so as to provide a secure location-based encryption scheme. Some of these conditions were as follows:

> *1. A message should be decryptable if and only if user is present at the location.*
>
> *2. A message should be decryptable only in one particular location area, where it is confined.*
>
> *3. A user should not be able to read an encrypted message for the second time without being there if he receives a new message at the same location.*

As can be seen from above conditions, a location-based encryption scheme is used so as to provide a solution to the elementary problem of how the location information could be included into the encryption scheme. Using existing cryptographic building blocks, we are not able to define geographical restrictions on the information. Hence, a secure protocol should find a way to integrate location data in a way that wouldn't allow a receiver to decrypt the message without being there at the correct location.

In this sense, our protocol uses location data as a means of authentication in the process of key derivation. While doing that, a certain range, called Tolerance Distance, is also included. Depending on the requirements, there should be also an option to extend the decryption area, as a desired location might occupy more than a single point. As shown in our protocol, we create quadrants where multiple positions fall into the same quadrant with respect to latitude and longitude so that depending on the tolerance distance, users can decrypt the messages. This prevents users who are located elsewhere to decrypt messages, as they wouldn't have the corresponding inputs for key derivation function.

When designing such a scheme, however, many problems start to arise, as location information in fact is not very suitable for this very task. The reason being is that location information lacks in something very important for the cryptographic building blocks called entropy. In cryptography, it is essential that an encryption key is computationally infeasible for an attacker to deduce information. Entropy, on the other hand, relates to this randomness of information which should be very hard for an attacker to guess. Now when we take a look at location data, as discussed in the previous chapters, vast areas of the Earth's surface have been covered with water sources. Furthermore, only a very small area of the solid area of the Earth has been urbanized. Under the circumstances, the randomness property of location information incredibly plummets. Moreover, a sender would leave the message at a reasonably close by area, as his/her acquaintances would also reside in this known area. Thus, this makes location information highly unsuitable for the encryption of sensitive information, as randomness property for the location information is very weak to be used directly in an encryption scheme which requires strong security. Lastly, using the location information directly as an encryption key is very troublesome because if the messages are encrypted with only the location data itself, the upcoming messages would also be revealed, as the same location would create the same encryption key each time. To prevent this, we use the Salt, unique per message, to create different encryption key for each message. This helps us a lot because by using different Salt per message, we prevent the creation of same key in case a receiver would try to guess the location and decrypt the message. This way, without the required Salt, the correct encryption key cannot be built. In addition to the Salt, we take advantage of the key derivation functions by increasing the iteration number, as this makes it computationally expensive. Hence, we diminish the problem of low entropy.

Above mentioned problems of location information are one of the main reasons in combination with verification of the location data, which will be covered in next sections, why location based encryption schemes are not used for a highly sensitive systems, such as banking and the military. While it is very important to ensure the security of the highly important systems, for other systems based on specific uses cases as we gave before, our protocol provides highly efficient protection to confine information into certain geographical area.

4.2.3 Location disclosure

In previous section, we talked about the security properties of location data. As mentioned before, if location data had directly used as an encryption key, an attacker would have directly brute force the input space or use precalculated tables to leverage small key space. Our countermeasure was to introduce computational expensiveness to slow down this processes as much as possible. In section 1.2, one of the most important properties of a secure location based application is stated as:

> *5. User's location should never be revealed or tampered with, e.g. to a server or to any other intermedi- aries.*

Our foremost concern in such a system is to retain location privacy. This is very crucial because as we showed in the introduction section, the loss of location privacy may result in certain consequences. When this is combined with a small key space, it creates a huge problem of retaining the privacy of a conversation from confidentiality stand point. Any information related to location data as well as auxiliary parameters helping in encryption should be kept secret, as knowing them would simplify further attack possibilities. To prevent this, our countermeasure is to use End-to-end Encryption (E2EE) to disable any intermediaries from tampering with the information. Based on public key cryptography, E2EE helps us to create a secure tunnel between two users during communication. This disables outsider factors from intercepting or eavesdropping the conversation. If an outsider attacker would be in the possession information such as, cipher-text, salt and initialization vector, he would have behaved like a receiver, which significantly enhances his chance of success during an attack.

However, even end-to-end encryption isn't necessarily the ultimate security solution which eliminates all the security problems. Instead of breaking the actual encryption, an attacker would disguise as a receiver. Thereafter, the messages coming from the sender are encrypted using his public key. Once he stores the messages, he could again encrypt the messages using the actual recipient's public key and send it to the receiver in order to not get caught. This is known as man-in-the-middle attack, but there are also tricks to prevent this , such as generating one time strings of characters based on users public keys [15]. In addition, no matter how strong the encryption keys are, the users mobile devices could get hacked and actual private keys could be stolen. However, we think under these circumstances end-to-end encryption appears to be the most secure way of handling data transmission, as many of the today's popular services are also rely on it for confidentiality purposes.

4.2.4 Location verification

In previous two sections, we looked at the security properties of the location data and the secrecy of communication channel. In this section, we will consider the reliability of the location data provided by a receiver from a verification standpoint.

When designing a location-based encryption protocol, one can think that the most important problem to answer is how we encrypt the message using location data. However, of all the problems we will encounter in this design process, how securely the protocol verifies the given location input will be the biggest one. As any location-based protocol will use the location data to encrypt the information, it is vital to verify location claims made by users. For example, with the help of fake GPS, the location data of a given device could be manipulated. Malicious users then could use this in their favor to provide fake location data in order to decrypt the given cipher-text without being at certain position they claim. Consequently, encrypting the messages using location data would not be safe, as any user could provide fake location data using this weakness. Many solutions have been proposed to prevent users from providing fake location data, such as using an additional location-authentication server to check the presence of nearby users. Alternatively, the nearby users could act as witnesses and create location proofs for each other using Bluetooth. Commonality of these approaches, however, is that they don't take location confidentiality into consideration, as other entities also do learn the location of an user for the sake of creating a proof.

Current design of our protocol does not make any verification for a given location input. Hence, fake location data would be provided to overcome the problem of traveling to a desired area, which creates a big problem. However, verification of location claims is a well-studied problem and requires another research; hence, we leave this part for another thesis study where verification of a given location input should be checked in order to prevent fake location claims. As location privacy is vital, an optimal solution should contain a location verification without any dissemination. In order to show how this can be done without any dissemination, we propose an improved version of LOCKUM in section 5.

5 Improved Version

In this section, we will present two countermeasures as improvements over the current design of LOCKUM. Both of the countermeasures aim at mitigating the problem of location verifications by adding additional parameters to key derivation function.

As we mentioned earlier, to verify the location claim of the users, we should find solutions that are not tampering with users' location privacy. For example, a location-authentication server could be used to create location proofs for the users of a given system. However, as location data are shared with another entity, the privacy requirements are jeopardized. Alternatively, nearby users with their smartphones could use Bluetooth communication with others to prove that a requester is present nearby. Both of these approaches where we have to share our location data with another entity contradict with our security needs, as we don't want to share our location information with someone else. Hence, we should focus on solutions where we could use environmental factors which prove that a user is present in a given location and can be added into our key derivation process.

Environmental factors through which we could verify the location claims of the users can be any data that are related to a nearby area, such as altitude data of a place. However, these factors should also possess certain properties. For example, in the case of using altitude, altitude data might be very easy to guess, as the altitude of a place will be in a very small set of numbers. Hence, a malicious user would again provide a lot of fake values to find the correct value by simply guessing this small set of values. In this sense, we suggest two approaches: using Media Access Control (MAC) address of a nearby Access Point(AP) and using a nearby cellular tower's CELL-ID.

5.1 Using MAC Address of a nearby Access Point(AP)

In this version, we aim at improving the credibility of the location data with an auxiliary parameter which is obtainable only if a receiver is physically located inside the desired area. We assume most of the encryption operations will take place in common places. These places typically consist of restaurants, cafes, museums, conference rooms, shopping malls and so on. Today, it is highly unconditional to find free Wireless Fidelity (WIFI) services provided by such places and most of the time users connect to these Local Area Networks (LANs) to leverage this opportunity.

Typically, WIFI Access Points (APs) reach the signals up to 45 meters indoors and 90 meters in open areas. This means anyone inside this range can discover and connect to these networks. However, if someone goes out of this range, the signal quality will plummet and will be most likely lost. Thus, this gives us an advantage of detecting people if they are actually inside this reachable area, as people who are far away from the WIFI AP won't be able to see the details of the network. If we take a look at WIFI APs, we can make use of the MAC address of the AP device. In essence, MAC Address of a device is a unique identifier assigned to a network interface controller (NIC) of the given network device. Accordingly, a person who is not inside this coverage area will not be able to know this particular information.

In general, MAC Addresses are all 48 bits in length, a few types of networks require 64-bit addresses instead, such as ZigBee. Besides that, they are represented in different forms such as:

> *MM:MM:MM:SS:SS:SS*
>
> *MM-MM-MM-SS-SS-SS*
>
> *MMM.MMM.SSS.SSS*

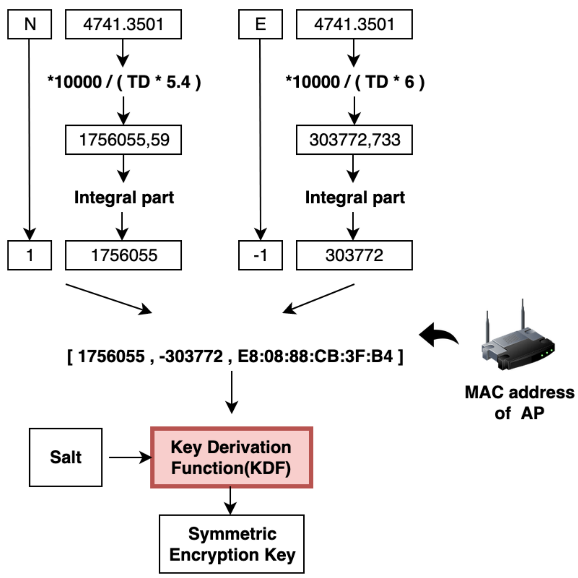

Figure 19: In this example, MAC address of the connected WIFI Access Point(AP)(E8:08:88:CB:3F:B4) is fetched by users as an additional parameter of the key derivation function to create a proof of existence in a given environment.

In Figure 19, it is shown how a MAC address of a WIFI AP is added as an additional parameter to key derivation function. This approach would be very beneficial to verify location claims of the users for use-cases mentioned in section 1.3, such as providing discount coupons inside a shopping center. Using this approach, a shop would verify that only the users inside the area would benefit from discount coupons compared to malicious users who use fake Global Positioning System (GPS) to target the location of the shopping center.

However, this approach would not be suitable at outdoor places where the devices in the given environment are using only cellular data. It aims at places with WIFI services; hence, users would have to be connected to an AP and create their corresponding input for the key derivation function for successful decryption. As a second problem, this approach creates verification for a location claim only for one time. For instance, a user would learn such information from a given environment, such as the MAC address of the AP. Next time, he would have a chance to use fake GPS with this information to provide a manipulated input without being at that location. Lastly, the signals from the AP source might be lost depending on the reachability of the AP, as each user would be located inside a given environment with a different distance to the AP.

In summary, using this approach, we can have an effective control mechanism for location verification of the users where the reachability and WIFI usage is not a problem for the connected devices.

5.2 Using a nearby cellular tower's CELL-ID

In the first improvement proposal, we used MAC address of an AP to add an additional layer of security from verification perspective. However, the majority of common areas aren't still perfectly covered with free WIFI services in addition to distance coverage problem of WIFI APs. As mentioned before, WIFI signals reach the signal up to 45 meters indoors and 90 meters in open areas. A typical mobile phone, on the other hand, has enough power to reach a cell tower signal from 20 to 70 kilometers distance. In this case, cellular data connection provides more reliable and complete coverage.

Indeed, reliability of the signals from the cell towers degrades as they go through concrete obstacles; hence, the signal reliability might be lower compared to WIFI services in indoor areas. Outdoor environments, on the other hand, are very suitable to use this method, as cell towers became omnipresent in today's cities. For example, when a mobile device is connected to a cellular tower, we can access to certain information about these towers and like we saw in the previous improvement proposal with WIFI APs, we can use certain information about this towers. Typically, a cellular tower has 4 identification fields:

- *MCC — a Mobile Country Code:* This code identifies the country. For example, 262 for Germany
- *MNC - a Mobile Network Code:* This code identifies the mobile operator. For example 11 for O2 (Germany) GmbH Co. OHG
- *LAC - a Location Area Code:* Location Area Code is a unique number of current location area. A location area consists of set of base stations.
- *CellID (CID) - an ID Number of a Cell Tower:* This number is unique number used to identify each cellular tower.

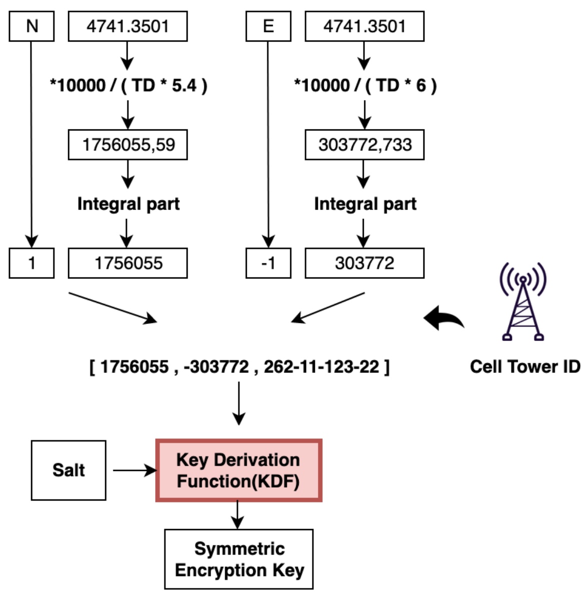

Figure 20: A set of ID's used to define a Cellular Tower is added into key derivation process, however, devices without General Packet Radio Service (GPRS) will not be able to use this function.

Using combination of these ID's defining an individual cell tower, we can verify that a mobile device is located inside the given area. Similar to previous proposal, we will have an ability to check whether or not a given device is located within this coverage area. However, not all of the devices are capable of using cellular data. For example, tablets with only WIFI capabilities are not able to communicate with these cellular towers. In addition to this, a cell ID is also one time solution for a verification problem, as an attacker might obtain this number once and use it again and again. Lastly, this approach would be more suitable for use cases where encryption is aimed for the outdoor areas, as indoor places block the transmission of cellular tower signals . Figure 20 shows an illustration of how the Cell ID of a cellular tower could be included into key derivation process.

In summary, both of these methods benefit from the environmental data which are related to their connection with certain network source. This provides a great practicality for controlling process. However, as the data used in this process are mostly static, an attacker would obtain this information once and would be able to use it again in future if he would try to attack to the system.

5.3 Security analysis of the improved version

In section 4.2, we made a security analysis of LOCKUM without any improvements. In this section, with the improved version, we suggested two environmental factors to diminish the problem of location verifications.

LOCKUM uses only latitude and longitude data during the key derivation process. However, as shown in section 4.2.4, the protocol lacks in during the verification of location claims. In this sense, the improved version diminishes the problem of this controlling. Using environmental factors, such as CELL-ID of a cell tower and a MAC address of the WIFI AP, helps us to see if an user is actually present at a location. However, we see that this information provides only one time verification, as the receiver can save this information and try to use it for the next time. Nevertheless, using this approach for certain use cases, we can ensure that the user was present at a location to decrypt the encrypted-text without using any other entity. This way, it is assured that confidentiality of the location information is still retained.

This contribution helps us to add an additional layer of security by changing the way key is constructed. Accordingly, while still providing the same level of security in terms of location disclosure and location authentication which are presented in section 4.2, we are able to improve location verification aspect of our protocol.

6 Future Work & Conclusion

6.1 Future work

Having introduced the protocol in previous chapters, we further suggested two improvements in the last chapter. As verification of location claims is one of the biggest problems with location-dependent encryption schemes, verifying location claims while still retaining the location privacy should be further investigated to prevent fake location data from being provided. An improved version should mitigate the problem of one-time verification which we explained in the improved version, as a receiver can cheat the system after the first verification. Environmental factors which could increase the entropy of the location data and can be added to key derivation function should be further investigated. Derivation of an encryption key based on location input could be further analysed and improved to prevent brute-force and rainbow-table attacks. As explained before, an iteration number is needed as an input to key derivation function. Encryption scheme should be implemented and tested as a mobile application to find the effects of the iteration number. Higher iteration numbers will yield in higher key derivation duration which would cause Denial-of-Service (DOS) attacks due to longer waiting periods. Besides, in the current version, a receiver may have to try adjacent quadrants to build a correct decryption key. This may result in up to 9 times slower decryption duration compared to an encryption operation. Hence, a new method should be further developed to facilitate efficient adjacent control to diminish this problem. Lastly, a new approach could be further developed to make it easier for a receiver to check whether or not he has the right encryption key after each time he builds a candidate decryption key.

6.2 Conclusion

In this master thesis, we presented LOCKUM, a new location-based encryption protocol which allows users to send messages that are geographically confined to a certain position while making sure that location privacy is not revealed to any other entity, to bring an additional layer of security dimension against the uprising problem of location privacy. Location Based Systems (LBSs) which provide functionality by actively capturing and using the location information of the users have been operating on a highly location-pervasive manner, as they can track users' location data constantly. Interestingly, this fact has long been ignored due to the fact that they provide great practicality for users. Today, we see that this pervasive manner of using location data of the users came to a point where it can cause severe results, such as obtaining confidential information regarding people's lives. Although the problem of retaining location-privacy appears to be an uprising problem, everyday users still seem to underestimate the importance of it. In our work, we presented a new location-based encryption protocol to prevent this problem and showed how location data can be used to create a cryptographically secure encryption key. We further analysed why location data itself can't be directly used to encrypt information due to its intrinsic lack of cryptographic entropy. As we mentioned earlier, the number of possible location points are limited due to the fact that the Earth's surface is mostly covered by water sources. In this sense, predictability of the location data jeopardizes the entire secrecy of location-based encryption methods, which could then create dramatic results. Accordingly, we explained what countermeasures are used to mitigate this problem of small encryption key space caused by location data and further touched on how the location privacy is not revealed while all this communication and cryptographic operations take place. Later, we defined our threat model where we showed structural vulnerabilities of our protocol. As we have shown there, auxiliary information which is used during an encryption operation to prevent certain attacks is securely transmitted through End-to-end Encryption (E2EE). Hence, the secrecy of the entire system relies on the security of E2EE, as knowing this information would incredibly simplify the things for an outsider attacker. Additionally, as shown in the security analysis, our protocol, unlike the improved version proposed earlier, doesn't verify location claims made by a receiver. Using this vulnerability, a malicious user can provide fake locations with the help of fake Global Positioning System (GPS). Hence, we suggest that encryption of highly sensitive data should not be subject to interpretation for the scope of this encryption protocol.

In summary, LOCKUM meets the security requirements that are needed to create a secure location-based encryption scheme by providing a complete design from the ground up for certain LBSs, such as location-based messaging applications, location-based games, and location-based advertisement systems. To extend the capabilities and target applications of LOCKUM, it should be further studied and improved by new approaches.

A Location Formats and Conversions

Location data refer to the position of an entity given a latitude and a longitude in a Cartesian coordinate grid. Generally speaking, one can, typically, encounter 3 different representations of the location data, such as **Degrees Minutes Seconds, Decimal Degrees, and Degrees Decimal Minutes**[18]. An example of the same location with three different representations can be shown as:

1. N 52° 31' 14.941" (Degrees Minutes Seconds)
2. 52.520817 (Decimal Degrees)
3. 5231.2490 (Degrees Decimal Minutes)

First one is the classical representation of the data in Degrees Minutes Seconds form. It could be interpreted as 52 Degrees 31 Minutes and 14.941 Seconds North. The second one is called Decimal Degrees, and they are used to show the latitude and the longitude values as decimal fractions and are used in many geographic systems. They are an alternative way to show degrees. The values are restricted by $\pm$ 90 and $\pm$ 180 for the latitude and the longitude, respectively. The last representation of the coordinates is named as Degrees Decimal Minutes. Here the value could be read as 52 Degrees 31.2490 Minutes. This representation bounds the latitude values from 0 to 89 and the longitude values from 0 to 179. For example 12134.234 would simply be read as 121 Degrees 34.234 Minutes whereas 934.111 would be read as 9 degrees 34.111 Minutes.

Essentially, most of our everyday mobile devices and global positioning tools show the location information in the form of Degrees Minutes and Seconds form or simply with Decimal Degrees. Thus, it is necessary to know how to convert each type of representation to another so that required positioning system would be converted. Here, we provide conversions regarding three common representations as follows:

A.1 Degrees Minutes Seconds to Decimal Degrees

The general rule regarding DMS(Degrees Minutes Seconds) to Decimal Degrees is as follows:

$$Decimal\ Degrees = Degrees + (Minutes/60) + (Seconds/3600)$$

An example conversion: N 52° 31' 14.941" (DMS)

$$Decimal\ Degrees = 52 + (31/60) + (14.941/3600) = 52.520817$$

A.2 Degrees Minutes Seconds to Degrees Decimal Minutes

The general rule regarding DMS(Degrees Minutes Seconds) to Degrees Decimal Minutes is as follows:

$$Degrees\ Decimal\ Minutes = Degrees + Minutes + (Seconds/60)$$

An example conversion: N 52° 31' 14.941" (DMS)

$$Degrees\ Decimal\ Minutes = 52 + 31 + (14.941/60) = 5231.2490$$

[18] more information at https://www.pgc.umn.edu/apps/convert/

A.3 Degrees Decimal Minutes to Decimal Degrees

The general rule regarding Degrees Decimal Minutes to Decimal Degrees is as follows:

$$Decimal\ Degrees = Degrees + [Minutes(IntegerPart) + (Minutes/60)(quotient)]$$

An example conversion: N 5231.2490 (Degrees Decimal Minutes)

$$Decimal\ Degrees = 52 + [31 + (31.2490/60)] = 5231.520817$$

B Haversine Distance Formula

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes[9]. Hence, we can use this formula to calculate what the corresponding latitude and the longitude values are for a 1 meter change. The formula is as follows:

$$Haversine\, a = sin^2(dLat/2) + sin^2(dLon/2).cos(lat1).cos(lat2)$$

$$c = 2arcsin(\sqrt{a})$$

$$d = R.c$$

Where,

- **d** is the distance between two points in kilometers
- **lat1, lat2** are the latitude values of first and second locations, respectively(in radians).
- **dLat, dLon** are the latitude and the longitude differences between two points, respectively(in radians).
- **R** is the radius of the Earth 6372.8 km (6372.8 km is an approximation of the radius of the average circumference (i.e., the average great-elliptic or great-circle radius), where the boundaries are the meridian (6367.45 km) and the equator (6378.14 km).)

Now, we can use the formula in order to find a 1 meter change for the latitude and the longitude values. Let us start by calculating 1 meter change for the *latitude*(we assume that two points are on the same longitude so that we can calculate the corresponding latitude change. Hence, *dLon* is 0):

$$d = R.c \tag{5}$$
$$0.001 = 6372.8 \times c \qquad \textbf{(c = 0,00000016)} \tag{6}$$
$$c = 2 \times (\arcsin\sqrt{a}) \tag{7}$$
$$a = sin^2(1/12500000) \tag{8}$$

Haversine a was:

$$Haversine\, a = sin^2(dLat/2) + sin^2(dLon/2).cos(lat1).cos(lat2)$$

if we calculate the equation,

$$a = sin^2(1/12500000) \tag{9}$$
$$sin^2(1/12500000) = sin^2(dLat/2) + sin^2(0/2).cos(lat1).cos(lat2) \tag{10}$$
$$sin^2(1/12500000) = sin^2(dLat/2) + 0 \tag{11}$$
$$dLat/2 = 1/12500000 \tag{12}$$
$$dLat = 1/6250000 = \mathbf{0.00000016} \tag{13}$$

We also know that this value is in radians and if we convert it into decimal degrees:

$$(DecimalDegrees\,x\,\pi)/180 = radians \tag{14}$$
$$(DecimalDegrees \times 3,14)/180 = 0.00000016 \tag{15}$$
$$\mathbf{0.0000091}\,Decimal\,Degrees \tag{16}$$

Lastly, we know from A.2 that decimal degree is (1/60) of the Decimal Minutes, thus:

$$(DecimalMinutes/60) = 0.0000091 \tag{17}$$
$$\textbf{Decimal Minutes} = 0.00054 \tag{18}$$

Finally, we know that we multiply the latitude and the longitude values with 10.000 so as to specify their precision; hence, also multiplying 0.00054 by 10000 to have them in same base gives us **5,4**. Thus, we say that 1 meter change in a given latitude value corresponds to 5,4.

We can now calculate the 1 meter corresponding change for a given longitude value assuming that they are on the same latitude(The Haversine formula requires a latitude value; hence, let us assume that we are located in Germany with the latitude of 47.683 degrees) :

$$d = R.c \tag{19}$$
$$0.001 = 6372.8.c \qquad \textbf{(c = 0,00000016)} \tag{20}$$
$$c = 2.(\arcsin\sqrt{a}) \tag{21}$$
$$a = sin^2(1/12500000) \tag{22}$$

Haversine a was:

$$Haversine\,a = sin^2(dLat/2) + sin^2(dLon/2).cos(lat1).cos(lat2)$$

if we calculate the equation,

$$a = sin^2(1/12500000) \tag{23}$$
$$sin^2(1/12500000) = 0 + sin^2(dLon/2).cos(0.83).cos(0.83) \tag{24}$$
$$1.9x10^{18} = sin^2(dLon/2)x0.99 \tag{25}$$
$$sin^2(dLon/2) = 1.9x10^{-18} \tag{26}$$
$$dLong = 0.00000016 \tag{27}$$

We also know that this value is in radians and if we convert it into decimal degrees:

$$(DecimalDegrees\,x\,\pi)/180 = radians \tag{28}$$
$$(DecimalDegrees \times 3,14)/180 = 0.00000016 \tag{29}$$
$$\mathbf{0.00000917}\,Decimal\,Degrees \tag{30}$$

Lastly, we know from A.2 that decimal degree is (1/60) of the Decimal Minutes, thus:

$$(DecimalMinutes/60) = 0.00000917 \tag{31}$$

$$\textbf{Decimal Minutes} = 0.0006 \tag{32}$$

Finally, we know that we multiply the latitude and longitude values with 10000 so as to specify their precision, hence, also multiplying 0.0006 by 10000 to have them in same base gives us **6** for this particular latitude but we know that the most of world's population lives between 0-45 latitude values so we can accept it as rounded to 6. Thus, we say that 1 meter change in a given latitude value corresponds to 6.

List of Abbreviations

AES Advanced Encryption Standart.

AP Access Point.

CSPNG Cryptographically Secure Pseudorandom Number Generator.

DES Data Encryption Standart.

DOS Denial-of-Service.

E2EE End-to-end Encryption.

GPRS General Packet Radio Service.

GPS Global Positioning System.

KDF Key Derivation Function.

LAN Local Area Network.

LBG Location Based Gaming.

LBS Location Based System.

LBSN Location Based Social Network.

MAC Media Access Control.

MMS Multimedia Messaging Service.

PBKDF2 Password Based Key Derivation Function 2.

PRNG Pseudo Random Number Generator.

SHA Secure Hash Algorithms.

SMS Short Message Service.

WIFI Wireless Fidelity.

Acknowledgements

This Master's Thesis would not have been possible without the contributions of many people, several of whom I cannot thank enough for their emotional support.

First of all, I am extremely grateful to my family for their continuous support. Obtaining a Masters Degree in Germany would be nothing but a dream without their help in this tiresome journey. I can't stress enough how much support they have given me, especially during the course of last year and a half, which has been incredible. This very achievement will always:DD be a source of pride for me and my family.

Many thanks to my fellow Turkish and other international friends, with whom I have spent almost 3 years in this amazing city, Konstanz. It was always pleasant to have many useful discussions with these valuable people to enlighten myself. Next, I would like to extend my sincere thanks to people from whom I got much support: Alper Ozaydin, Nida Cilasun, Mahmut Ozer and Selcuk Cabuk.

I would also like to send my sincere thanks to Formigas GmbH team, with whom I spent almost a year. Especially, Jens Hensler and Martin Liersch, whose supervising skills were very helpful for me.

Lastly, I would like to thank my supervisor Klaus Herberth. His guidance has taught me how to cope with various problems and overcome them during my master's degree. Also, I am very thankful for the long discussions which helped me a lot during my thesis.

References

[1] Ala Al-Fuqaha and Omar Al-Ibrahim. Geo-encryption protocol for mobile networks. *Computer Communications*, 30(11-12):2510–2517, 2007.

[2] The Earth Institute at Columbia University. The Growing Urbanization of the World. `https://www.earth.columbia.edu/news/2005/story03-07-05.html`, 2005. [Online; accessed 23-May-2019].

[3] Alastair R Beresford and Frank Stajano. Mix zones: User privacy in location-aware services. In *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second*, pages 127–131. IEEE, 2004.

[4] Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir. Key recovery attacks of practical complexity on aes-256 variants with up to 10 rounds. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 299–319. Springer, 2010.

[5] Giacomo Brambilla, Michele Amoretti, and Francesco Zanichelli. Using blockchain for peer-to-peer proof-of-location. *arXiv preprint arXiv:1607.00174*, 2016.

[6] Dominik Bucher, David Rudi, and René Buffat. Captcha your location proof—a novel method for passive location proofs in adversarial environments. In *LBS 2018: 14th International Conference on Location Based Services*, pages 269–291. Springer, 2018.

[7] Laurent Bussard and Walid Bagga. Distance-bounding proof of knowledge to avoid real-time attacks. In *IFIP International Information Security Conference*, pages 223–238. Springer, 2005.

[8] Michael Chan. Mobile phones and the good life: Examining the relationships among mobile use, social capital and subjective well-being. *New Media & Society*, 17(1):96–113, 2015.

[9] Nitin R Chopde and M Nichat. Landmark based shortest path detection by using a* and haversine formula. *International Journal of Innovative Research in Computer and Communication Engineering*, 1(2):298–302, 2013.

[10] David D Clark. The end-to-end argument and application design: the role of trust. *Fed. Comm. LJ*, 63:357, 2010.

[11] Ksenia Ermoshina, Francesca Musiani, and Harry Halpin. End-to-end encrypted messaging protocols: An overview. In *International Conference on Internet Science*, pages 244–254. Springer, 2016.

[12] Levent Ertaul, Manpreet Kaur, and Venkata Arun Kumar R Gudise. Implementation and performance analysis of pbkdf2, bcrypt, scrypt algorithms. In *Proceedings of the International Conference on Wireless Networks (ICWN)*, page 66. The Steering Committee of The World Congress in Computer Science, 2016.

[13] Antonio M Espinoza, William J Tolley, Jedidiah R Crandall, Masashi Crete-Nishihata, and Andrew Hilts. Alice and bob, who the {FOCI} are they?: Analysis of end-to-end encryption in the {LINE} messaging application. In *7th {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 17)*, 2017.

[14] Joao Ferreira and Miguel L Pardal. Witness-based location proofs for mobile devices. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–4. IEEE, 2018.

[15] Andy Greenberg. Hacker lexicon: What is end-to-end encryption? *Ηλεκτρονικό]. Available: https://www. wired. com/2014/11/hacker-lexicon-end-to-end-encryption/.[Πρόσβαση Οκτώβριος 2017]*, 2014.

[16] Hatem Hamad and Souhir El Kourd. Key strength with encryption and dynamic location of mobile phone. In *2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, pages 468–473. IEEE, 2012.

[17] Hatem Hamad and Souhir Elkourd. Data encryption using the dynamic location and speed of mobile node. *Journal of Media and Communication Studies*, 2(3):067–075, 2010.

[18] Michael Herrmann. *Privacy in Location-Based Services*. PhD thesis, PhD thesis. Leuven, Belgium: KU Leuven–Faculty of Engineering Science, 2016.

[19] Amir Herzberg and Hemi Leibowitz. Can johnny finally encrypt?: evaluating e2e-encryption in popular im applications. In *ACM Workshop on Socio-Technical Aspects in Security and Trust (STAST)*, 2016.

[20] Haosheng Huang, Georg Gartner, Jukka M Krisp, Martin Raubal, and Nico Van de Weghe. Location based services: ongoing evolution and research agenda. *Journal of Location Based Services*, 12(2):63–93, 2018.

[21] Chitra Javali, Girish Revadigar, Kasper B Rasmussen, Wen Hu, and Sanjay Jha. I am alice, i was in wonderland: secure location proof generation and verification protocol. In *2016 IEEE 41st conference on local computer networks (LCN)*, pages 477–485. IEEE, 2016.

[22] Yaron Kanza. Location corroborations by mobile devices without traces. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 60. ACM, 2016.

[23] Nadim Kobeissi. An analysis of the protonmail cryptographic architecture. Cryptology ePrint Archive, Report 2018/1121, 2018. `https://eprint.iacr.org/2018/1121`.

[24] Gaëtan Leurent and Thomas Peyrin. From collisions to chosen-prefix collisions application to full sha-1. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 527–555. Springer, 2019.

[25] Hsien-Chou Liao and Yun-Hsiang Chao. A new data encryption algorithm based on the location of mobile users. *Information Technology Journal*, 7(1):63–69, 2008.

[26] Hsien-Chou Liao, Po-Ching Lee, Yun-Hsiang Chao, and Chin-Ling Chen. A location-dependent data encryption approach for enhancing mobile information system security. In *The 9th International Conference on Advanced Communication Technology*, volume 1, pages 625–628. IEEE, 2007.

[27] Xinye Lin and Wenbo He. Wilove: A wifi-coverage based location verification system in lbs. *Procedia Computer Science*, 34:484–491, 2014.

[28] Helger Lipmaa, Phillip Rogaway, and David Wagner. Ctr-mode encryption. In *First NIST Workshop on Modes of Operation*, volume 39. Citeseer, 2000.

[29] Wanying Luo and Urs Hengartner. Veriplace: a privacy-aware location proof architecture. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 23–32. ACM, 2010.

[30] Liu Mengjun, Liu Shubo, Zhang Rui, Li Yongkai, Wang Jun, and Cao Hui. Privacy-preserving distributed location proof generating system. *China Communications*, 13(3):203–218, 2016.

[31] Logan Scott and Dorothy E Denning. A location based encryption technique and some of its applications. In *ION National Technical Meeting*, volume 2003, pages 730–740, 2003.

[32] Mainak Sen, Sachin Tibrewal, Sumana Majumder, Nehal Ahmad, and Kishlay Kumar Singh. Location dependent cryptographic algorithm based on open source application. *International Journal of Computer Applications*, 975:8887, 2017.

[33] Ahren Studer and Adrian Perrig. Mobile user location-specific encryption (mule): using your office as your password. In *Proceedings of the third ACM conference on Wireless network security*, pages 151–162. ACM, 2010.

[34] Jawahar Thakur and Nagesh Kumar. Des, aes and blowfish: Symmetric key cryptography algorithms simulation based performance analysis. *International journal of emerging technology and advanced engineering*, 1(2):6–12, 2011.

[35] Meltem Sönmez Turan, Elaine Barker, William Burr, and Lily Chen. Recommendation for password-based key derivation. *NIST special publication*, 800:132, 2010.

[36] Xinlei Wang, Amit Pande, Jindan Zhu, and Prasant Mohapatra. Stamp: Enabling privacy-preserving location proofs for mobile users. *IEEE/ACM transactions on networking*, 24(6):3276–3289, 2016.

[37] Ming Yan, Chien Aun Chan, Wenwen Li, I Chih-Lin, Sen Bian, André F Gygax, Christopher Leckie, Kerry Hinton, Elaine Wong, and Ampalavanapillai Nirmalathas. Network energy consumption assessment of conventional mobile services and over-the-top instant messaging applications. *IEEE Journal on Selected Areas in Communications*, 34(12):3168–3180, 2016.

[38] Lin You, Yulei Chen, Bin Yan, and Meng Zhan. A novel location-based encryption model using fuzzy vault scheme. *Soft Computing*, pages 1–11, 2018.