

Privacy-Preserving Proximity Detection through Haversine Distance and Geo-Hash

by

Jash Rathi
202011070

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY

in

INFORMATION AND COMMUNICATION TECHNOLOGY

to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

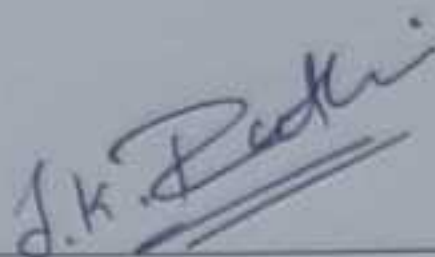


May, 2022

Declaration

I hereby declare that

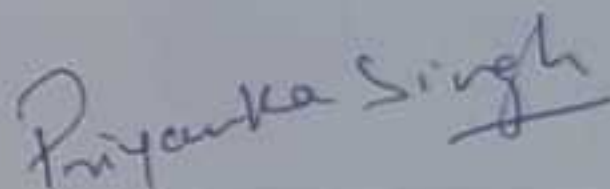
- i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,
- ii) due acknowledgment has been made in the text to all the reference material used.



Jash Rathi

Certificate

This is to certify that the thesis work entitled "Privacy-Preserving Proximity Detection through Haversine Distance and Geo-Hash" has been carried out by Jash Rathi for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my / our supervision.



Prof. Priyanka Singh
Thesis Supervisor

Declaration

I hereby declare that

- i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,
- ii) due acknowledgment has been made in the text to all the reference material used.

Jash Rathi

Certificate

This is to certify that the thesis work entitled "Privacy-Preserving Proximity Detection through Haversine Distance and Geo-Hash" has been carried out by Jash Rathi for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my /our supervision.

Prof. Priyanka Singh
Thesis Supervisor

Acknowledgments

It is very well said one should never stop learning. During my MTech journey of two years, I learned how to handle real-life problems in the computer science world. Given this beautiful opportunity, I would like to thank the Dhirubhai Ambani Institute of Information and Communication Technology. Thank you, god, for this great opportunity and a fantastic experience.

Foremost, I would like to express my sincere gratitude to my guide Prof. Priyanka Singh who has been a constant support and motivation for me. Her knowledge in the research is immense, which helped me throughout my work. I could not have imagined a better guide with so much patience for my MTech thesis.

I am also thankful to my fellow research colleague and friend Nimmi Patel for being a solid support system. She had always boosted me from refining drafts to removing errors when I struggled.

At last, I would thank my family, professors, and friends for giving their precious time to build me into a good human. I appreciate the time each of you has spent helping me in any situation. I will always be grateful and will never forget your teaching.

Contents

Abstract	iv
List of Tables	v
List of Figures	vi
1 Introduction	1
1.1 Introduction	1
1.2 Organization	4
2 Preliminaries	5
3 Related work	7
4 Proposed Method	11
4.1 Threat Model	11
4.2 Proximity detection through Haversine distance (P3-HD)	12
4.3 Proximity detection through Geohash (P3-GH)	15
4.4 Secure anomaly detection using homomorphic encryption (SADHE)	17
4.5 Proposed Algorithms	20
5 Implementation and Results	24
5.1 Dataset	24
5.2 Experimental details of P3-GH and P3-HD	24
5.3 Experimental details of SADHE	26
6 Security Analysis	29
7 Conclusion	32
References	33

Abstract

Proximity implies computing suitable distance between nearby people and places. Proximity detection is very crucial to many multimedia applications. For instance, fatal incidents occur near forests if animals like elephants walk into nearby crowded regions, resulting in loss of life and property. During the COVID-19 pandemic, the infection spread globally due to a lack of proper proximity detection strategies. Many solutions were proposed based on Bluetooth, WiFi, wearable sensors, ultrasound, and GPS co-location to detect the proximity. However, privacy-preserving solutions were limited, which hindered the privacy-aware society. In this paper, we propose two privacy-preserving proximity detection frameworks called Privacy-Preserving Proximity through Haversine Distance (P3-HD) and Privacy-Preserving Proximity through Geohash (P3-GH). With the help of available GPS data and the traditional Haversine formula, P3-HD computes the distance to detect if the user is in the proximity zone. The other framework, P3-GH detects the proximity on the top of the encrypted Geohashes. These proposed frameworks secure the data of the authentic user from any unauthorized access. GPS data may introduce anomalies that lead to false results. We have proposed a secure framework to detect the anomalies called SADHE, a secure method for detecting anomalies in GPS trajectory without compromising users' location. We compare various primacy and deficiency against potential threats, specifically replay attacks, poison attacks, frequency analysis attacks, and man-in-the-middle attacks, and validate the robustness of the proposed frameworks.

List of Tables

3.1	Primacy and deficiency of the existing proximity detection schemes	10
5.1	Comparison of proximity points for P3-GH (9 bits) & P3-HD ($d \leq 3$ m)	25
5.2	Comparison of proximity points for P3-GH (8 bits) & P3-HD ($d \leq 19$ m)	26
5.3	Comparison of existing method [11] and SADHE	26

List of Figures

4.1	The architecture of Privacy-Preserving Proximity Detection through Haversine Distance (P3-HD)	12
4.2	Connection establishment phase	13
4.3	The architecture of Privacy-Preserving Proximity Detection through Geohash (P3-GH)	15
4.4	The architecture of Secure Anomaly Detection using Homomorphic Encryption (SADHE)	17
5.1	Comparison of P3-GH (9 bits) and P3-HD ($d \leq 3m$)	24
5.2	Comparison of P3-GH (8 bits) and P3-HD ($d \leq 19m$)	25
5.3	Comparison of existing method [11] and SADHE	26
5.4	Distance computed using traditional Haversine and proposed Haversine formula	27
5.5	Velocity computed using traditional Haversine and proposed Haversine formula	27
5.6	Acceleration computed using traditional Haversine and proposed Haversine formula	28
6.1	Defense against replay attack	29
6.2	Defense against poison attack	30

CHAPTER 1

Introduction

1.1 Introduction

Proximity implies computing the suitable distance between nearby people and places. In general, proximity does not feed its absolute location, nor does it give a solution for going there. Proximity issues become so popular during the COVID-19 pandemics. The infection spread globally due to the infected people lying in the vicinity of the healthy ones. To combat the spread of COVID-19, medical professionals suggested maintaining a safe distance. Proximity detection and digital contact tracing became popular during the pandemics. Several multimedia exist in the real world based on proximity detection schemes. With the help of multimedia tools and technologies, human daily life activities can be monitored easily. It includes primarily medical and pharma sectors, unmanned aerial vehicles (UAVs), and robotics. Detecting transmissible infectious diseases is one of the emerging areas in which proximity detection strategies are used. Nowadays, artificial intelligence and machine learning algorithms analyze human behavior and social interaction. Autonomous vehicles (AV) and robotic cars are rapidly growing research work areas in which proximity detection strategies can be used. Several fatal accidents happen due to the proximity of autonomous vehicles, which result in losing lives, properties, and equipment involved in accidents. Several proximity detection schemes can avoid fatal incidents, such as video cameras. But cameras do not remain reliable under poor light and bad weather. Cameras can also be blinded by dirt, dust, and fog. Another way to avoid fatal incidents is to use a radar system. The radar system recognizes objects present in its range. Sometimes it triggers false alarms due to moving debris, even birds and animals. The system with a false alarm suffers from a lot of nuisance. These events can be avoided by alerting the user with the proximity detection system.

Most of the solutions are based on Bluetooth, WiFi, wearable sensors, ultrasound, and GPS co-location to detect the proximity. Bluetooth devices can detect the proximity in its short range. Bluetooth signal strength gets reduced as the distance increases. The Bluetooth signal gets absorbed by the surrounding objects. That is another reason Bluetooth devices fail to detect proximity. WiFi technology is also used to detect proximity. The majority of the devices that use WiFi technology suffers from privacy issues. The user's privacy gets compromised when they use publicly available WiFi. Public WiFi enables unethical users to sniff/eavesdrop, session hijacking, etc. GPS is also a widely used technology to detect short-range and long-range proximity. But accessing the GPS data can reveal the user's location, frequently visited places, habits, political involvements, spiritual aptitude, etc. Due to these privacy issues, GPS and WiFi-based detection schemes are not accepted by society. So these demands for privacy-preserving proximity detection approaches.

In most cases, GPS provides accurate results. But sometimes, it gives false GPS readings, leading to anomalies in GPS data. Anomalies may lead to wrong decisions and inaccurate results that create problems in availing LBS. We have proposed a framework called secure anomaly detection using homomorphic encryption (SADHE).

In this work, we have proposed two proximity detection frameworks called Privacy-Preserving Proximity through Haversine Distance (**P3-HD**) and Privacy-Preserving Proximity through Geohash (**P3-GH**). We assume mainly three entities in the proposed framework: user, CSP, and adversary. In the proposed framework P3-HD, the user will provide his location-based information to the CSP to stay secure from the proximity zone. The CSP will use the GPS data and the traditional Haversine formula to calculate the distance. The CSP will detect whether the user is in the proximity zone based on the distance. Unauthorized users try to gain information through an insecure communication channel and threaten the user's privacy. By encrypting the GPS data user's identity remain hidden, and no unauthorized user can access the data. In the proposed framework P3-GH, the user sent his encrypted Geohashes to the CSP. Then on the top of the encrypted data, the CSP will detect whether the user is free from the proximity zone. In both these approaches, we detect the proximity while preserving the user's data privacy. We have conducted various experiments to validate the robustness of the proposed frameworks.

The major contributions of the work are as follows:

1. **Detecting the proximity while preserving the user's privacy:** The proposed frameworks can detect the proximity without revealing the user's actual locations. The user's identity remains anonymous due to the encrypted data. In the proposed framework, P3-HD, the user sends the encrypted GPS coordinates to the CSP. The CSP will calculate the distance using the Haversine formula. In P3-GH, the user provides encrypted Geohashes to the CSP. Based on data available from other active users and data received from the actual user, it will verify and detect the proximity. If the user is in a proximity zone, it will alert the user.
2. **Maintaining confidentiality:** Information privacy is referred to as confidentiality. When data is transmitted over an insecure communication channel, an external adversary can intercept the communication. The proposed framework employs homomorphic encryption to resolve the issues above. The user sends the encrypted information to preserve the user's privacy. So even if the data is intercepted, it does not compromise confidentiality.
3. **Preserving the data integrity:** Though the user sends the encrypted GPS data to the CSP, it can be tampered intentionally or unintentionally by an adversary to deteriorate the services. The proposed scheme ensures that data integrity must be preserved against these malicious activities by verifying the received GPS data.
4. **Anomaly detection in GPS data:** Sometimes, signal distortion, bad weather, bad antenna position, poor mobile networks, etc., may introduce anomalies in GPS reading that give inaccurate results. Before detecting the proximity, we ensure that GPS data must be accurate. Inaccurate GPS data introduces anomalies that lead to wrong decisions and create problems in availing location-based services. We have proposed SADHE to detect the anomalies based on the distance, velocity, and acceleration parameters and remove them to clean the trajectory before outsourcing.
5. **Defense against potential attacks:** The insecure communication channel opens the door for an adversary to access the underlying information. Adversaries also gain information through attacks like man-in-the-middle attacks and frequency analysis attacks. We have conducted various experiments to validate the robustness of proposed methods against potential at-

tacks, specifically, replay attacks, poison attacks, frequency analysis attacks, and man-in-the-middle attacks.

1.2 Organization

The work is organized as follows: Chapter 2, 3 presents preliminaries and related work respectively. Chapter 4,5 presents the proposed frameworks and it's results respectively. We compare various primacy and deficiency in the security analysis chapter 6, and finally, the conclusion.

CHAPTER 2

Preliminaries

This chapter discusses potential attack scenarios: replay attack, poison attack, frequency analysis attack, and man-in-the-middle attack.

- **Replay attack:** A replay attack is a passive attack in which an unauthorized user does not know ongoing communication. He observes the pattern of messages and captures some of these messages. After some time, the attacker forwards these messages to the original destination pretending to be an authorized user.
- **Poison attack:** This attack is combination of erasure attack and duplicate faking attack. Threat occurs in a system when the data stored on the server is different from the actual data. Either the source of data has got erased or the stored data has been modified. Therefore failing the data integrity check.
 - Erasure attack For data M , whose tag value is T an adversary store cipher text C' instead of C on server. Now if the legitimate client tries uploading same data M with actual cipher C on sever, request will be rejected. As to upload data M client will pass the tag value T , server will check T in its log. On server as T is already stored mapping to cipher C' the actual data upload request is not accepted.
 - Duplicate faking attack The attack occurs when client fails to detect the modified data during download phase, and hence accepts the corrupted data. This attack can be stopped by performing data integrity check on client side before the download phase.
- **Frequency analysis attack:** A frequency analysis attack is based on monitoring the frequency of messages. The attacker observes the ciphertext, and if the same ciphertext is sent multiple times, he can map that it corresponds to the same original message. For instance, consider a scenario in which the ciphertext C is sent that corresponds to the original message M . If an attacker

observes that ciphertext C is sent multiple times, the attacker can map that ciphertext C belongs to the same message M .

- **Man-in-the-middle attack:** A man-in-the-middle attack is a cyber-attack where an unauthorized user intercepts the ongoing communication between two authenticated users without their awareness. This attack is possible when communication between two users occurs over an unencrypted channel.

CHAPTER 3

Related work

The majority of the solutions use Bluetooth, WiFi, wearable sensors, radar, and GPS colocation to detect the proximity as shown in Table 3.1. Carreras et al. proposed proximity detection through smartphones [2]. They proposed the mechanism called "comm2Sense", in which proximity gets detected through WiFi, WiFi hotspots, and WiFi receivers. They achieved a proximity range of 0-5 meters. Dmitrienko et al. proposed a proximity detection framework based on WiFi colocation [5]. They proposed a framework for digital contact tracking using scanning and storing information. Deterministic if/else classifier was used to detect the proximity from WiFi scan data and hotspot duty cycle. But as discussed earlier public WiFi opens the door for an adversary to steal sensitive data and fails to preserve privacy. WiFi-based devices suffer from several privacy issues, including cyber-attacks, session hijacking, packet sniffing/eavesdropping, etc.

Social behavior analysis is also one of the rapidly growing research work areas. Today with advanced sensing technology, it is easy to monitor human interactions and social behavior. Choudhury et al. proposed sensor-based human interactions, proximity, and social network analysis [4]. They offered a "Sociometer," a wearable sensor that measures group interactions. A hidden Markov model is used to learn the pattern of IR signals. Several drawbacks of technical tools like sensors and chips are comparatively costlier than other proximity detection schemes. Also, it requires daily and heavy maintenance.

Cattuto et al. proposed a framework that allows monitoring social interactions and identifying the community connector patterns [3]. The idea is to identify proximity based on face-to-face interactions. It is based on the Radio Frequency Identification (RFID) devices that evaluate mutual proximity by exchanging low-power radio packets. Krumm et al. proposed a "NearMe" solution without absolute location [6]. NearMe compares clients' lists of WiFi access points and signal strengths to compute the proximity of devices. NearMe can give relative distance for short-range proximity without absolute locations.

Leith et al. proposed a framework for Coronavirus detection using Bluetooth LE signal [7]. They report a Bluetooth LE received signal measurement in various environments (i.e., in a meeting, train carriage, and grocery shop). They noticed during their experiments that Bluetooth signal strength varied from device to device, and the signal strength gets reduced as the distance increases. Also, Bluetooth devices fail to detect the long-range proximity. The Bluetooth signal gets absorbed by the surrounding objects like walls, clothes, cupboards, etc.

Nieto et al. proposed proximity warning system (PWS) and transmission locking mechanism [9]. It is based on mobile equipment and GPS location to avoid vehicle accidents. Picco et al. offer wildscope system whose key functionality is geo-referenced proximity detection of an animal to others or to landmarks based on GPS signals and GSM modem. GPS data is compensated data [12]. GPS gadgets can reveal the user's identity and expose personal and religious habits, political involvements, etc. Hence, exposure of the GPS data can threaten the user's privacy and reveal a person's identity. This is a limiting factor for the GPS gadgets that data privacy must be preserved while using them. Marks et al. proposed a framework for proximity detection and alert technology for safe construction equipment operation [8]. This framework is based on alerting the operators through Personal protection unit (PPU) and Equipment Protection Units (EPU) devices. It gives a warning when Objects are near humans' safe zone.

GPS is a contemporary solution used to move from one place to another. In most cases, GPS provides accurate results. Sometimes, signal distortion, bad weather, bad antenna position, poor mobile networks, etc., may introduce an anomaly in GPS reading and give inaccurate results. Anomalies may lead to wrong decisions and create problems in availing the LBS services. Several solutions have been proposed to detect anomalies from GPS data while achieving user data privacy. Patil et al. proposed an anomaly detection technique based on statistical data analysis [11]. The author proposed a framework to detect the outliers using the z-test method. To compute the distance, they have modified the actual Haversine formula. Using the modified Haversine formula, they reduce the computational cost. Sari et al. have discussed the implication of anomaly detection systems in cloud environments, their types, techniques, and the restrictions of each method [13]. In this work, they have compared the security measures of Dropbox, Google Drive, and iCloud using the AES encryption algorithm. Barucija et al. proposed a solution to detect the outliers based on statistic data analysis [1]. Patil et al. proposed an anomaly detection technique based on statistical data analysis [11]. The author proposed a framework to detect the outliers using the

z-test method. To compute the distance, they have modified the actual Haversine formula to reduce the computational cost.

These existing solutions require Bluetooth, WiFi, advanced sensors, high maintenance chips, technical tools, etc. That is why it is not cost affordable to everyone. Also, this tool requires daily and heavy maintenance. As discussed earlier majority of the existing system can detect the proximity but fail to preserve privacy. User identity will remain hidden in the proposed frameworks, and data privacy will also be preserved. We have used an encryption scheme to hide the user's actual location. Using P3-HD and P3-GH, users' location never gets disclosed to unauthorized users. Also, the proposed architecture does not need any complex hardware devices, high-cost sensors, etc. Also, there is no direct interaction between participating users.

Table 3.1: Primacy and deficiency of the existing proximity detection schemes

Technology	Benefits	Limitations
Bluetooth	Low initial cost Minimum infrastructure needed	Not suited for long range Signal absorption by objects
WiFi	Low initial cost	Not suited for long range Privacy issues, session hijacking, packet sniffing/eavesdropping, etc.
Laser	High accuracy	High initial cost Only applicable for short range Can't differentiate between human and objects
Video camera	Differentiate between human and objects	High maintenance cost Poor work ability in dusty, foggy area
Sonar	Minimum infrastructure needed	Can not differentiate between human and objects
Magnetic fields	Differentiate between human and objects	High battery power
GPS	Detect short-range and long-range proximity	Ability to reveal the location

CHAPTER 4

Proposed Method

This chapter gives a detailed description of the proposed frameworks P3-HD, P3-GH, and SADHE. It also discusses the involved entities and their specific roles.

4.1 Threat Model

This subsection presents the details of the involved entities. There are mainly three entities in this model: user, CSP, and adversary. The communication channel between the user and the CSP is considered as insecure channel.

- **User:** A user is an entity that preserves the safe distance in his surrounding to other humans or objects. To detect whether a user holds the proximity, it provides encrypted GPS data to the CSP. A user is assumed to be an honest entity.
- **CSP:** The CSP receives the encrypted GPS data sent by the user and detects whether the user is free from the proximity zone or not. The CSP detects the proximity based on the user's location and its database values. Its database contains other active users' location-based information surrounding the actual user. It periodically maintains the database by dumping the old data to check the proximity with the recent data. The CSP is assumed to be a semi-honest entity.
- **Adversary:** An adversary intercepts the ongoing communication between the user and the CSP and gets to know the information exchanged. It is considered to be a malicious entity.

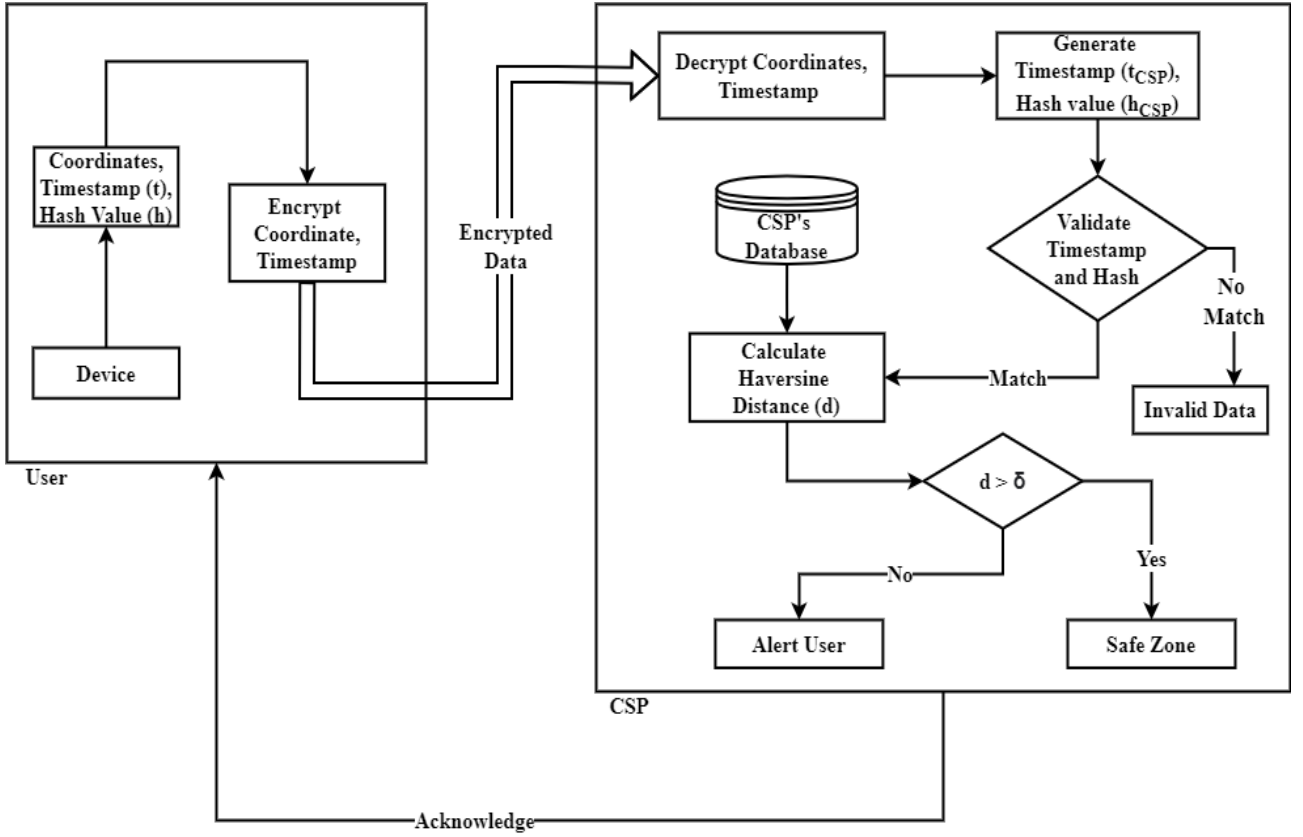


Figure 4.1: The architecture of Privacy-Preserving Proximity Detection through Haversine Distance (P3-HD)

4.2 Proximity detection through Haversine distance (P3-HD)

In this subsection, we provide all the steps of the proposed framework P3-HD as depicted in Fig. 4.1.

User:

Step 1: The user provides the GPS data to the CSP. GPS data consists of latitude, longitude, and timestamp value. Before encryption, every pair of latitude and longitude is multiplied by a scalar value by 10^6 to extract the integer portion.

Step 2: Then, the user encrypts the extracted integer values of latitude and longitude using the Paillier homomorphic encryption [10]. Let the extracted integer values of latitude and longitude be $lat = \{lat_1, lat_2, \dots, lat_n\}$ and $long = \{long_1, long_2, \dots, long_n\}$ respectively. Each of these integer values

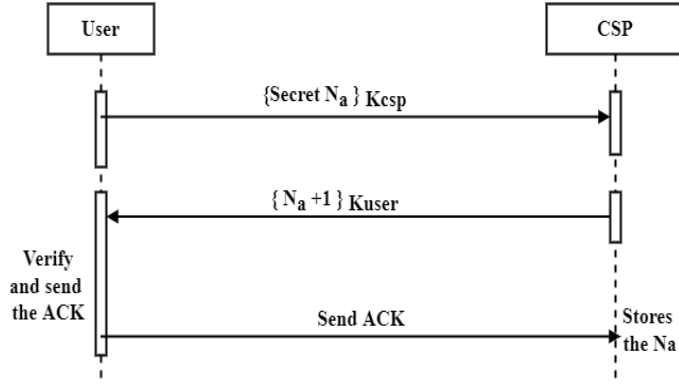


Figure 4.2: Connection establishment phase

are encrypted independently as shown in algorithm 1. At the same time, the timestamp t of the recorded GPS data is also encrypted as e_t .

$$e_{lat_i} = E(lat_i) \quad (4.1)$$

$$e_{long_i} = E(long_i) \quad (4.2)$$

$$e_{t_i} = E(t_i) \quad (4.3)$$

where, lat_i and e_{lat_i} represent the i^{th} value of latitude and corresponding encrypted value of latitude. $long_i$ and e_{long_i} represent the i^{th} value of longitude and corresponding encrypted value of longitude. $E()$ is the encryption function. This gives us encrypted values of latitude and longitude as $e_{lat} = \{e_{lat_1}, e_{lat_2}, \dots, e_{lat_n}\}$ and $e_{long} = \{e_{long_1}, e_{long_2}, \dots, e_{long_n}\}$ respectively.

Step 3: The user also computes the hash value of the extracted integer values of the latitude as follows:

$$h_i = H((lat_i - lat_{i-1}) \oplus N_a) \quad (4.4)$$

where, h_i represents the hash value of the i^{th} data point in the trajectory. $H()$ is the hash function, N_a represents the secret random nonce known only to the authorized user and the CSP as shown in Fig. 4.2.

The user repeats steps 1 to 3 for every consecutive pair of the GPS trajectory. Thereafter, he sends the entire set of encrypted data e_{lat}, e_{long} , the

encrypted timestamp e_t and the hash value h to the CSP.

CSP :

The CSP is assumed to be an honest entity in P3-HD because it needs to decrypt the GPS data to detect the proximity.

Step 4: The CSP receives the encrypted data from the user and decrypts it as follows:

$$lat_i = D(e_{lat_i}) \quad (4.5)$$

$$long_i = D(e_{long_i}) \quad (4.6)$$

$$t_i = D(e_{t_i}) \quad (4.7)$$

where, D is the decryption function.

Step 5: After decryption, the CSP will compute the hash value h_{CSP} as shown in algorithm 3. To check whether data integrity is preserved or not, it will compare h_{CSP} with the hash value h sent by the user.

Step 6: Also, CSP checks for any delayed messages based on the decrypted timestamp value d_t .

Step 7: Once the validation phase is over (data integrity check in step 5 and delayed messages in step 6), the CSP will compare user coordinates and the data stored in its database to detect the proximity. It uses the traditional Haversine Eq. 4.10 to calculate the distance d between two points as shown in algorithm 4.

$$a = \sin\left(\frac{lat_2 - lat_1}{2}\right)^2 + \cos(lat_1) * \cos(lat_2) + \sin\left(\frac{lang_2 - lang_1}{2}\right)^2 \quad (4.8)$$

$$c = 2 * \text{atan2}(\sqrt{a}, \sqrt{1 - a}) \quad (4.9)$$

$$d = r * c \quad (4.10)$$

where r is the radius of the earth that is 6,371 km, and d is the distance.

Step 8: Based on the calculated distance, we will alert the user if two users do not maintain a safe distance.

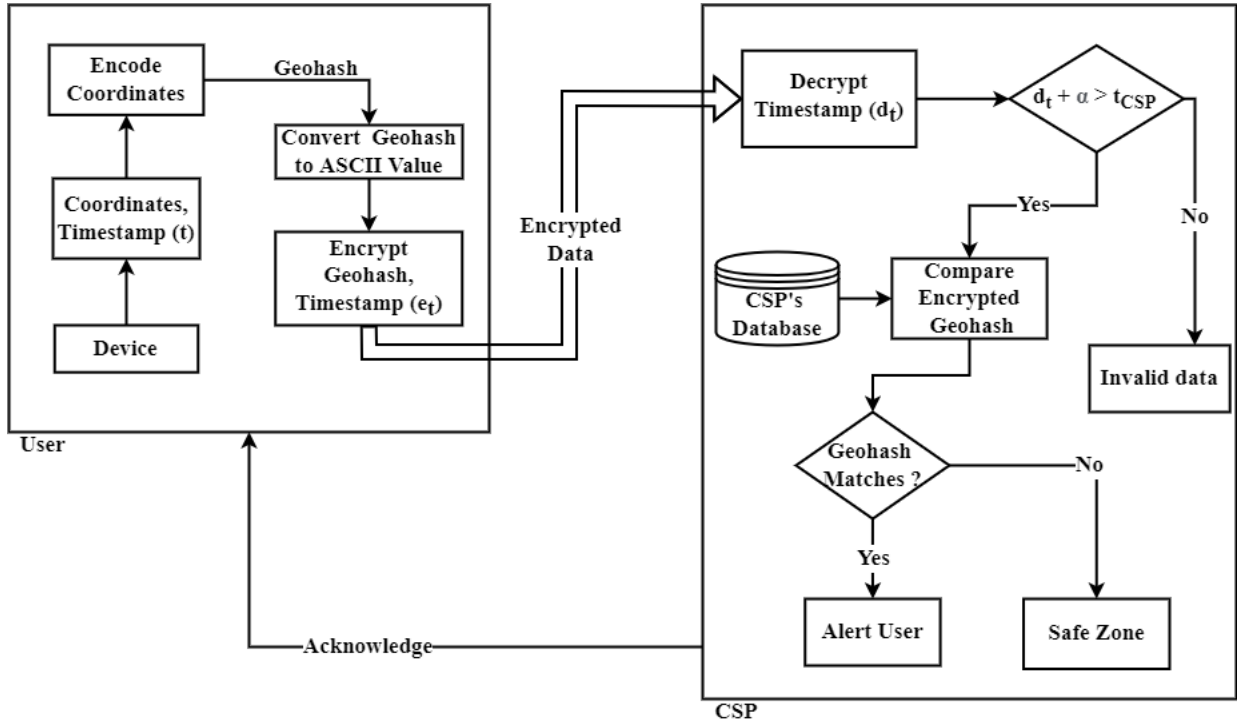


Figure 4.3: The architecture of Privacy-Preserving Proximity Detection through Geohash (P3-GH)

4.3 Proximity detection through Geohash (P3-GH)

In this subsection, we provide all the steps of the proposed framework P3-GH as depicted in Fig. 4.3.

User:

Step 1: The user's location can be represented in terms of GPS coordinates. The coordinates are a pair of latitudes and longitudes representing the user's exact location.

Let the values of latitude and longitude be $lat = \{lat_1, lat_2, \dots, lat_n\}$ and $long = \{long_1, long_2, \dots, long_n\}$ respectively.

Step 2: The user will encode each pair of coordinates and generate the Geohash [14]. Geohash is an alphanumeric string containing numeric values and

alphabet characters. Geohash generation process will be followed as shown in algorithm 5.

Step 3: Encrypting the non-numeric value is complex compared to numeric values. Each character in Geohash is replaced with its ASCII value to encrypt it.

Step 4: After substitution, the user has encrypted the Geohash value using Paillier homomorphic encryption algorithm. It also encrypts the timestamp value t corresponding to the Geohash. The user sends the encrypted data to the CSP using an insecure communication channel.

$$e_{gh_i} = E(gh_i) \quad (4.11)$$

$$e_{t_i} = E(t_i) \quad (4.12)$$

where, gh_i and e_{gh_i} represent the i^{th} value of Geohash and corresponding encrypted value of Geohash. t_i and e_{t_i} represent the i^{th} value of timestamp and corresponding encrypted value of timestamp. $E()$ is the encryption function.

CSP:

Step 5: The CSP receives the encrypted data from the user and verifies the data based on timestamp conditions as shown in Fig. 6.1. It decrypts the timestamp value as follows:

$$t_i = D(e_{t_i}) \quad (4.13)$$

where, D is the decryption function.

Step 6: Once the validation phase is over (delayed messages in step-5), the CSP compares the user's Geohash data with the data stored in its database. It will use the timestamp value to check the recent data in its database. It stores the location-based information (i.e. Geohashes) of other active user's surrounding the actual user. The CSP periodically maintains the database by dumping the old data.

Step 7: On the top of the encrypted data, using the paillier additive property, the CSP will compare the encrypted Geohashes to detect the proximity. If the user is found to be in a proximity zone, then the CSP will alert the user.

4.4 Secure anomaly detection using homomorphic encryption (SADHE)

This subsection gives a detailed description of the proposed method SADHE, a secure anomaly detection scheme using homomorphic encryption as depicted in Fig. 4.4.

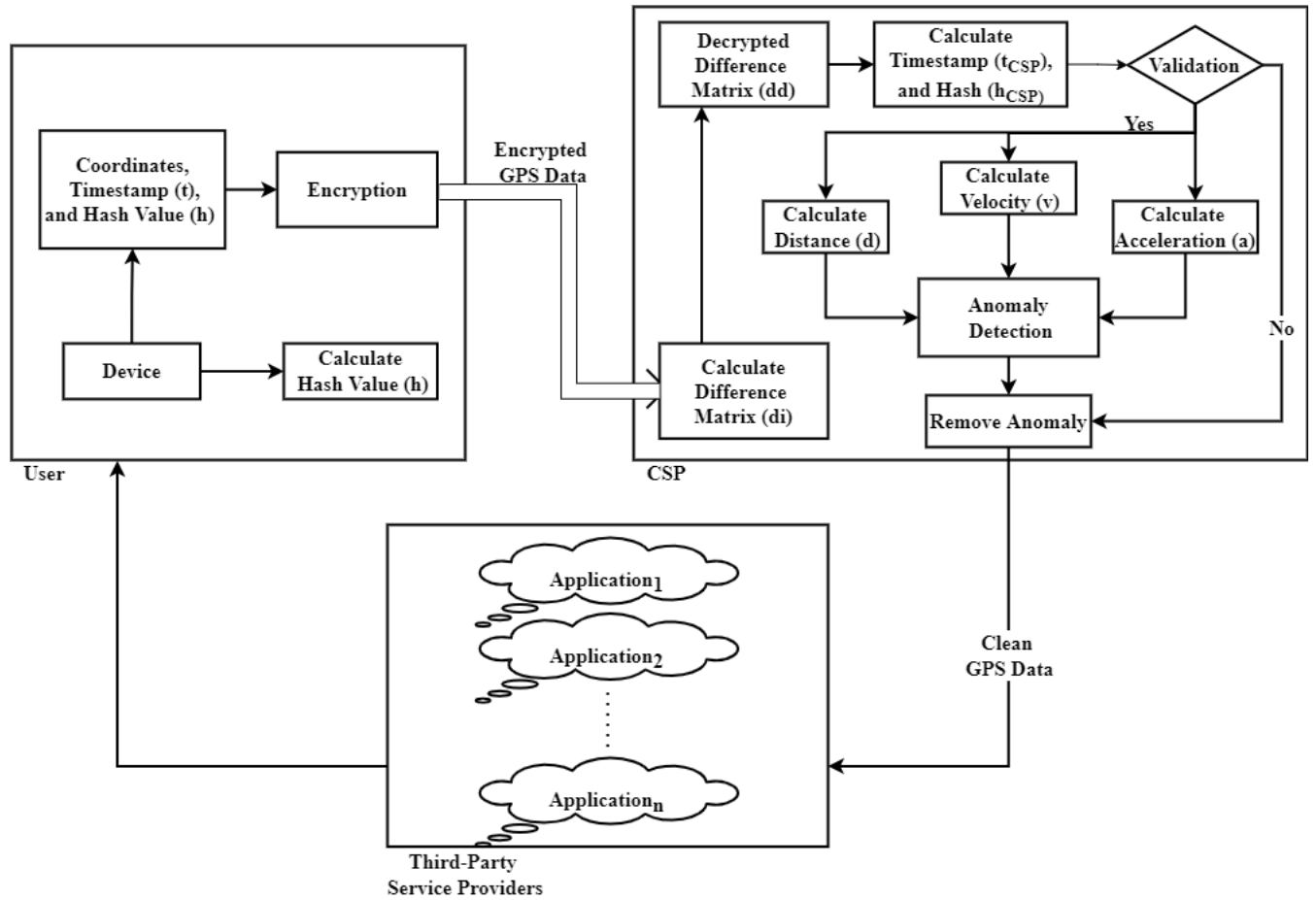


Figure 4.4: The architecture of Secure Anomaly Detection using Homomorphic Encryption (SADHE)

User:

Step 1: The user provides the encrypted GPS data to the CSP. GPS data consists of latitude, longitude, and timestamp value. Before encryption, every pair of latitude and longitude is multiplied by a scalar value, by 10^6 to extract the integer portion.

Step 2: Then the user encrypts the extracted integer values of latitude, longitude and the timestamp value.

Step 3: The user also computes the hash value of the extracted integer values of the latitude as Eq. 4.4:

The user repeats steps 1 to 3 for every consecutive pair of the GPS trajectory. Thereafter, he sends the entire matrix of encrypted data e_{lat}, e_{long} , the encrypted timestamp e_t and the hash value h corresponding to the GPS trajectory to the CSP.

CSP :

Step 4: The CSP receives the data from the user and calculates the difference between consecutive pairs of encrypted latitude and longitude to generate the difference matrix di as follows:

$$d_{lat_i} = D(e_{lat_i}, e_{lat_{i-1}}) \quad (4.14)$$

$$d_{long_i} = D(e_{long(i)}, e_{long(i-1)}) \quad (4.15)$$

where, d_{lat_i} and d_{long_i} represent the columns of i^{th} difference value of adjacent latitude and longitude. $D()$ is the difference function.

Step 5: The CSP decrypts the difference matrix. The CSP computes hash value h_{CSP} for the decrypted difference matrix. CSP compares the computed hash value h_{CSP} with the hash value h sent by the user. If both the hash values match, then the data integrity of the GPS data is preserved.

Step 6: Also, CSP checks for any delayed messages based on the decrypted timestamp value d_t as shown in Fig. 6.1.

Step 7: After finishing the validation phase (data integrity check in step-5 and delayed messages in step-6), the CSP uses the modified Haversine Eq. 4.18 to calculate the distance d between two consecutive points.

According to the small-angle approximation theorem, we have approximated the values of the trigonometric functions assuming the angle between two successive coordinates is very small (i.e., $\sin\Theta \approx \Theta$). Based on the literature [14], we can approximate the product of the $\cos(lat1)$ and $\cos(lat2)$ to be 1. Based on these modifications, we have derived the modified Haversine formula as follows:

$$a = \left(\frac{lat_2 - lat_1}{2} \right)^2 + \left(\frac{long_2 - long_1}{2} \right)^2 \quad (4.16)$$

$$c = 2 * atan2(\sqrt{a}, \sqrt{(1 - a)}) \quad (4.17)$$

$$d = r * c \quad (4.18)$$

Step 8: Based on calculated distance d , the CSP will calculate the velocity v and acceleration a . CSP will also calculate mean values of distance, velocity, and acceleration as \bar{d} , \bar{v} , and \bar{a} .

Step 9: The CSP will calculate the threshold value based on the frequency distribution of distance. In our case, the threshold values is considered as 5. The threshold value is calculated as follows:

$$\delta = \left(\frac{\text{frequency range of distance}}{\text{mean value of distance}} \right) \quad (4.19)$$

Step 10: Calculate the anomalies for every parameter distance d , velocity v , and acceleration a , and store them in the separate anomaly vectors as follows:

$$\vec{D} = \forall i \in d (d_i > \delta * \bar{d}) \quad (4.20)$$

$$\vec{V} = \forall i \in v (v_i > \delta * \bar{v}) \quad (4.21)$$

$$\vec{A} = \forall i \in a (a_i > \delta * \bar{a}) \quad (4.22)$$

Step 11: Thereafter, the final anomalies are computed based on the intersection of the corresponding vectors as follows:

$$\vec{O} = \{\forall i \in ((\vec{D}_i \cap \vec{V}_i) \cup (\vec{V}_i \cap \vec{A}_i) \cup (\vec{A}_i \cap \vec{D}_i))\} \quad (4.23)$$

Step 12: Once identified, these anomalies are removed to obtain the cleaned data and sent to the third-party service providers.

Third-party service providers:

Step 13: Third-party service providers like Google maps, Ola, Uber, Swiggy, and Zomato will need clean GPS data to calculate the distance between source and destination without knowing users' actual locations.

4.5 Proposed Algorithms

This subsection represents the algorithms for the proposed frameworks.

Algorithm 1 P3-HD

The user will encrypt the GPS data using Paillier homomorphic encryption.

INPUT: Set of latitude, longitude, and timestamp

OUTPUT: Encrypted values of latitude, longitude, and timestamp

Define: 1. $lat = \{lat_1, lat_2, \dots, lat_n\}$ is the set of latitude

2. $long = \{long_1, long_2, \dots, long_n\}$ is the set of longitude

3. $t = \{t_1, t_2, \dots, t_n\}$ is the set of timestamp value

4. $e_{lat} = \{e_{lat_1}, e_{lat_2}, \dots, e_{lat_n}\}$ is the set of encrypted latitudes

5. $e_{long} = \{e_{long_1}, e_{long_2}, \dots, e_{long_n}\}$ is the set of encrypted longitude

6. e_t is encrypted timestamp value

7. N_a is secret random nonce

8. $h = \{h_1, h_2, \dots, h_n\}$ is the set of hash value

9. $E()$: It will encrypt the data using Paillier encryption algorithm.

10. $H()$: It will generate the hash value.

1: **procedure** ENCRYPT($lat, long$)

2: **for** i **do** in $lat, long$

3: $i \leftarrow i * 10^6$

4: **end for**

5: **for** i **do** in $lat, long, t$

6: $e_{lat_i}, e_{long_i}, e_{t_i} \leftarrow E(lat_i, long_i, t_i)$ ▷ (Encryption of GPS data)

7: **end for**

8: **for** i **do** in lat

9: $h_i \leftarrow H(\{(lat_i - lat_{i-1}) \oplus N_a\})$ ▷ (Calculate hash value)

10: **end for** ▷ (Send encrypted data to CSP)

11: **end procedure**

Algorithm 2 P3-HD

The CSP will decrypt the data sent by the user

INPUT: Encrypted values of latitude, longitude, and timestamp

OUTPUT: Decrypted values of latitude, longitude, and timestamp

Define: 1. $D()$: It will decrypt the encrypted data sent by the user.

```
1: procedure DECRYPT( $e_{lat}, e_{long}$ )
2:   for  $i$  do in  $e_{lat}, e_{long}$ 
3:      $lat, long \leftarrow D(e_{lat}, e_{long})$            ▷ (Decryption of GPS coordinates)
4:   end for
5:   for  $i$  do in  $t$ 
6:      $t_i \leftarrow D(e_{t_i})$                        ▷ (Decryption of timestamp value)
7:   end for
8: end procedure
```

Algorithm 3 P3-HD

The CSP will compute the hash value and verify the data.

INPUT: Decrypted set of latitude, longitude, and timestamp

OUTPUT: Valid set of latitude, longitude, and timestamp

Define: 1. h_{CSP} is hash value computed by the the CSP.

```
1: procedure VALIDATE( $h_{CSP}, lat$ )
2:   for  $i$  do in  $h_{CSP}$ 
3:      $h_{CSP_i} \leftarrow H(\{(lat_i - lat_{i-1}) \oplus N_a\})$    ▷ (CSP will Calculate the hash
value)
4:   end for
5:   for  $i$  do in  $h_{CSP}, h$ 
6:     if  $h == h_{CSP}$  then
7:       Valid data
8:     end if                                           ▷ (Data integrity validation)
9:   end for
10: end procedure
```

Algorithm 4 P3-HD

The CSP will detect the proximity based on the Haversine distance

INPUT: Set of latitude, longitude and timestamp

OUTPUT: Calculated Havesine distance

Define: 1. r is a radius of the earth (i.e., 6435 km)

2. d is a distance calculated using traditional Haversine formula.

```
1: procedure DIST( $lat, long$ )
2:   for  $i$  do in  $lat, long$ 
3:     for  $j$  do in  $lat, long$ 
4:        $a = \sin\left(\frac{lat_j - lat_i}{2}\right)^2 + \cos(lat_i) * \cos(lat_j) + \sin\left(\frac{lang_j - lang_i}{2}\right)^2$ 
5:        $c = 2 * \text{atan2}(\sqrt{a}, \sqrt{1 - a})$       ▷ Haversine distance calculation
6:        $d = r * c$ 
7:       if  $d < 3m$  then
8:         proximity occurs                      ▷ Alert the user
9:       end if
10:    end for
11:  end for
12: end procedure
```

Algorithm 5 P3-GH :

The user will encrypt the Geohashes using Paillier homomorphic encryption

INPUT: Set of latitude, longitude and timestamp

OUTPUT: Encrypted set of Geohashes

Define: 1. gh is Geohash value

2. e_{gh} is encrypted Geohash value

3. $E()$: It will encrypt the data using Paillier encryption algorithm.

4. $ENCODE()$: It will encode the GPS coordinates to Geohash

```
1: procedure ENCRYPTION( $lat, long$ )
2:   for  $i$  do in  $lat, long$ 
3:      $i \leftarrow i * 10^6$ 
4:   end for
5:   for  $i$  do in  $lat, long$ 
6:      $gh_i \leftarrow ENCODE(lat_i, long_i)$       ▷ (Geohash generation)
7:      $e_{gh_i} \leftarrow E(gh_i)$                 ▷ (Encryption of Geohash)
8:   end for
9:   for  $i$  do in  $t$ 
10:     $e_{t_i} \leftarrow E(t_i)$                   ▷ (Encryption of timestamp value)
11:  end for                                     ▷ (Send encrypted data to CSP)
12: end procedure
```

Algorithm 6 P3-GH :

The CSP will detect the proximity on the top of the encrypted data

INPUT: Encrypted set of Geohashes

OUTPUT: Acknowledge the user

Define: 1. gh is Geohash value

2. e_{gh} is encrypted Geohash value

3. $D()$: It will decrypt the timestamp using Paillier encryption algorithm.

1: **procedure** PROXIMITY

2: **for** i **do** in e_t

3: $t_i \leftarrow D(e_t)$

▷ Validation phase

4: **end for**

5: **for** i **do** in e_{gh_i}

6: **for** j **do** in e_{gh_j}

7: **if** $i=j$ **then**

8: Alert the user

▷ Check for the proximity

9: **end if**

10: **end for**

11: **end for**

12: **end procedure**

CHAPTER 5

Implementation and Results

This chapter presents the details of our experiments to validate the proposed frameworks.

5.1 Dataset

We have used the Microsoft Geolife dataset to examine our proposed frameworks [15]. We have used a few trajectories, which consist of around 21000 points. The dataset contains the parameters like latitudes, longitudes, it's recorded time-tamp, etc.

5.2 Experimental details of P3-GH and P3-HD

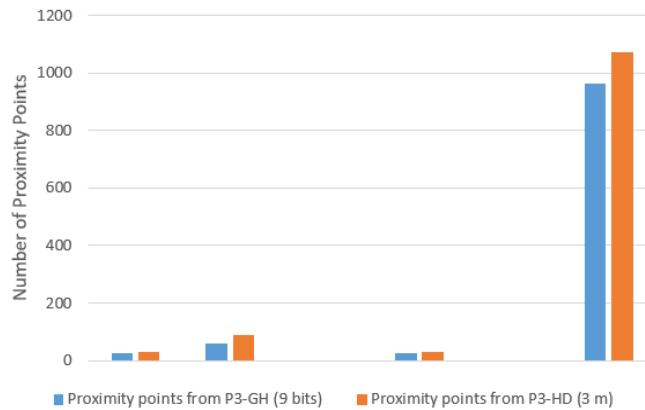


Figure 5.1: Comparison of P3-GH (9 bits) and P3-HD ($d \leq 3m$)

The proposed framework P3-HD is based on the available GPS data and the traditional Haversine formula. The CSP will use this GPS data to calculate the distance. Based on the distance, the CSP will detect the proximity. The proposed framework P3-GH is based on the encrypted Geohashes. We have taken the first

Table 5.1: Comparison of proximity points for P3-GH (9 bits) & P3-HD ($d \leq 3$ m)

Text File (Total points)	P3-GH (9 bits) (proximity points)	P3-HD($d \leq 3$ m) (proximity points)
005_708, 005_230 (4500)	27	29
001_104, 001_305 (3000)	61	88
020_301, 020_907 (6700)	0	0
020_825, 020_926 (2300)	24	28
020_301, 020_443 (3300)	0	0
002_523, 002_805 (6700)	965	1071
Total points(21000)	1077	1216

few bits from the entire twelve-bit string of the Geohash value. There are several proximity chances if it matches with any other string stored in the CSP'S database. We compared these two frameworks on the bunch of GPS data to analyze the accuracy.

We performed experiments with the proposed frameworks on the Geolife datasets and got almost same results. We got 828, and 992 proximity points out of twenty-one thousand points for P3-GH with the first 9 bits and P3-HD with the Haversine distance less than 3 meters, respectively. These results are shown in Fig. 5.1 and Table 5.1 respectively.

We got 1077, and 1216 proximity points out of twenty-one thousand points for P3-GH with the first 8 bits and P3-HD with the Haversine distance less than 18 meters, respectively. These results are shown in Fig. 5.2 and Table 5.2 respectively. In terms of privacy, P3-GH is more suitable than P3-HD because it does not need to decrypt the data. P3-GH is able to detect the proximity on the top of the encrypted data without compromising the user's data privacy.

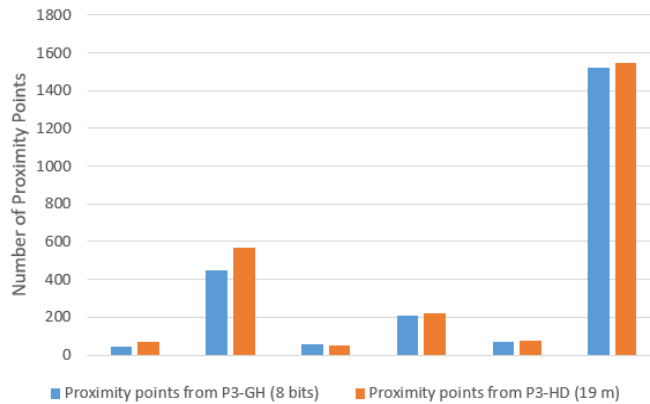


Figure 5.2: Comparison of P3-GH (8 bits) and P3-HD ($d \leq 19$ m)

Table 5.2: Comparison of proximity points for P3-GH (8 bits) & P3-HD ($d \leq 19$ m)

Text File (Total points)	P3-GH (8 bits) (proximity points)	P3-HD ($d \leq 19$ m) (proximity points)
005_708, 005_230 (4500)	47	72
001_104, 001_305 (3000)	450	571
020_301, 020_907 (8000)	49	56
020_825, 020_926 (2300)	208	223
020_301, 020_443 (3300)	67	77
002_523, 002_805 (6700)	1523	1549
Total points(21000)	2344	2548

5.3 Experimental details of SADHE

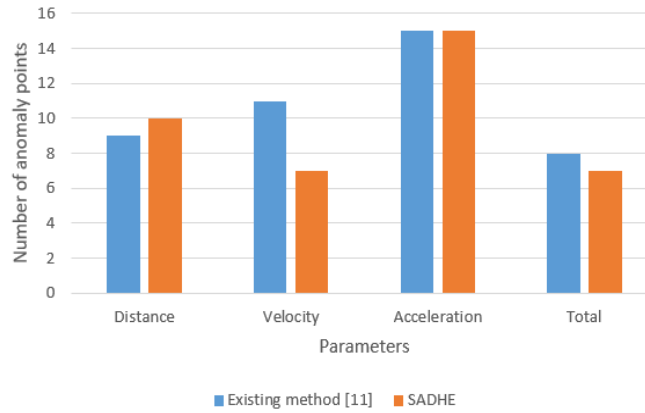


Figure 5.3: Comparison of existing method [11] and SADHE

Table 5.3: Comparison of existing method [11] and SADHE

Parameters (Anomaly counts)	Existing method [11]	SADHE
Distance	9	10
Velocity	11	7
Acceleration	15	15
Final count	8	7

We have computed anomaly points using the z-test method [11] and the proposed SADHE method. We achieved almost similar anomaly points for the z-test method [11] and the proposed SADHE method. We have shown the results for one of the trajectories consisting around one thousand points as shown in Fig. 5.3. For the z-test method, we got 9, 11, and 15 anomaly points based on the distance, velocity, and acceleration parameters. The proposed SADHE has 8, 7, and

15, respectively. The final anomaly points came out to be 8 and 7 for the z-test method and the SADHE, respectively.

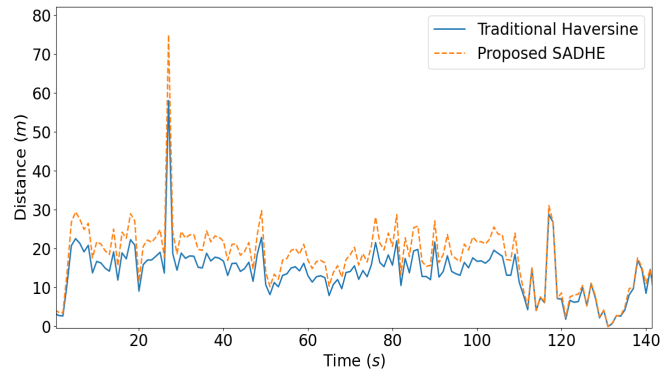


Figure 5.4: Distance computed using traditional Haversine and proposed Haversine formula

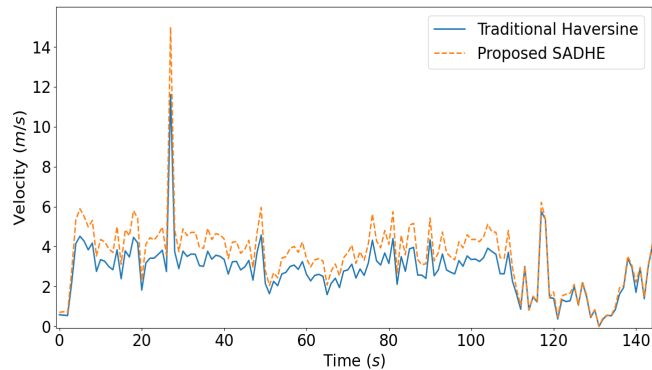


Figure 5.5: Velocity computed using traditional Haversine and proposed Haversine formula

We have calculated the distance, velocity, and acceleration parameters using the traditional Haversine Eq. 4.2 and modified Haversine Eq. 4.18 to check the accuracy of the modified Haversine formula. We got approximately same calculations using modified Haversine formula in compared to traditional Haversine formula as shown in Fig. 5.4, 5.5, and 5.6.

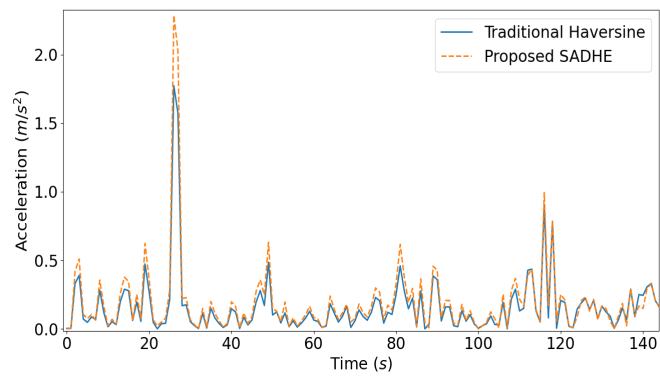


Figure 5.6: Acceleration computed using traditional Haversine and proposed Haversine formula

CHAPTER 6

Security Analysis

In this chapter, we have analyzed the security of the proposed frameworks P3-HD and P3-GH, and SADHE.

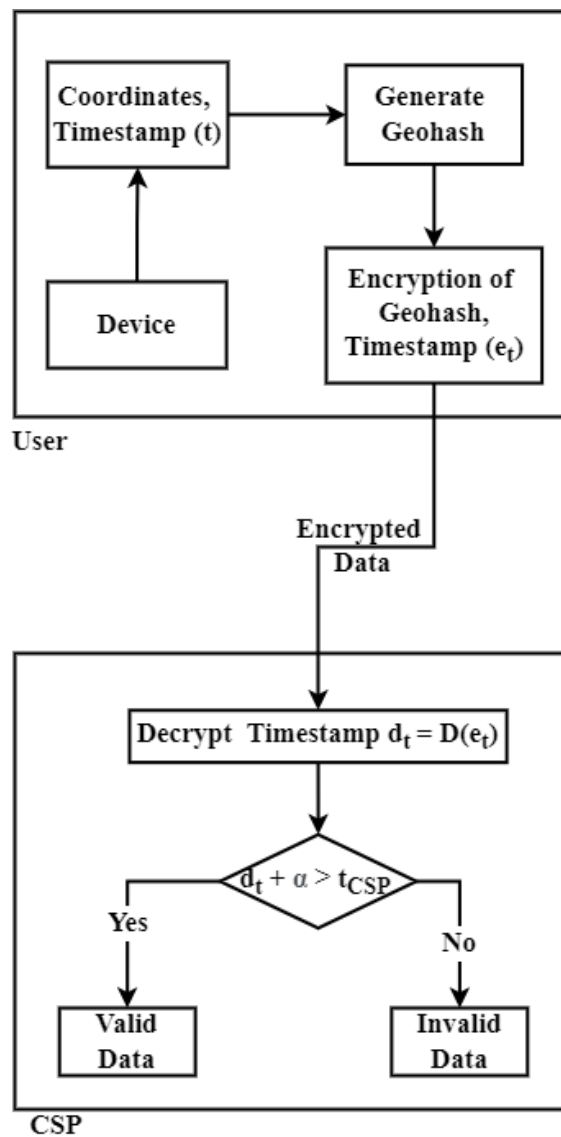


Figure 6.1: Defense against replay attack

Lemma 6.0.1. : *The proposed frameworks are secure against replay attacks.*

Proof. We are using the timestamp-based defense mechanism to detect replay attacks. The user sends the encrypted timestamp e_t of the coordinates along with the encrypted coordinates to the CSP. The CSP will decrypt the timestamp value d_t received from the user. Based on the decrypted timestamp value d_t , the CSP will validate the data. If $d_t + \alpha > t_{CSP}$, then the received message is considered as valid otherwise it gets discarded. Here, α and t_{CSP} represent the threshold value of the timestamp and current timestamp value of the CSP, respectively. In our experiments, we have set the α value to be 180 seconds. Therefore, the proposed frameworks are secure against replay attacks. \square

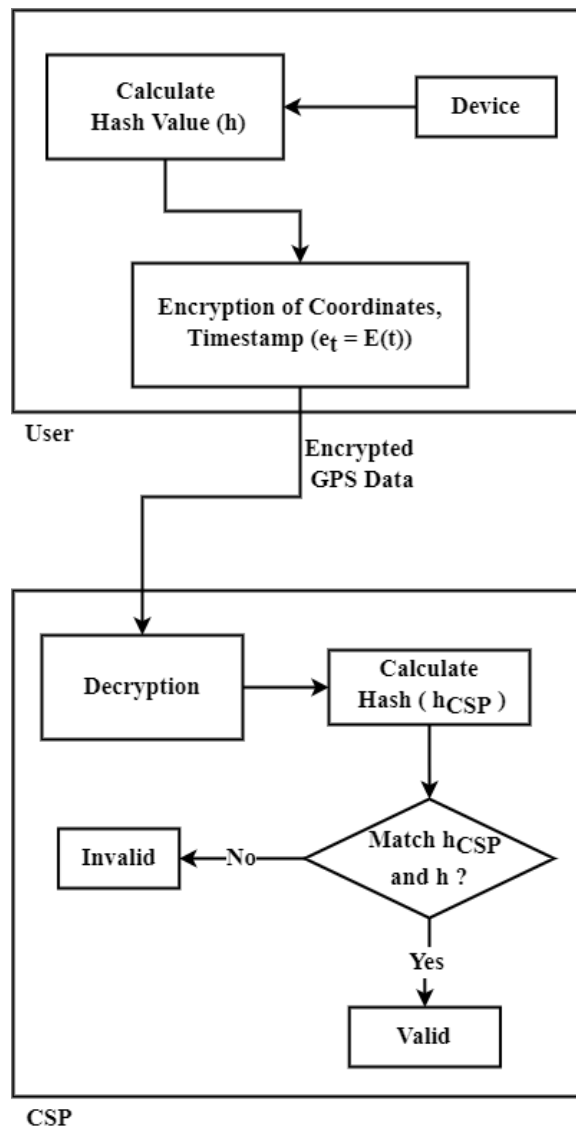


Figure 6.2: Defense against poison attack

Lemma 6.0.2. : *The proposed frameworks are secure against poison attacks.*

Proof. A poison attack combines an erasure attack and a duplicate faking attack. In an erasure attack, the original data on the server is removed by the adversary. In a duplicate faking attack, the original data on the server is replaced with some malicious data. In both cases, the authentic user loses the actual data, which results in a violation of data integrity. P3-HD uses robust hashing using Eq. 4.4 to detect whether data integrity is preserved or not as shown in Fig. 6.2. The user sends the hash value h of the coordinates and the encrypted coordinates to the CSP. The CSP will calculate the hash value h_{CSP} based on the data it receives and then verify it with the hash value sent by the user. By verifying the hash value, the CSP verifies whether data integrity is preserved or not. Therefore, the proposed frameworks are secure against poison attacks. □

Lemma 6.0.3. : *The proposed frameworks are secure against frequency analysis attacks.*

Proof. We use the Paillier homomorphic encryption, a probabilistic scheme to prevent the frequency analysis attack. Even for the same plaintext, we get different ciphertexts. It avoids any direct mapping between the plaintext-ciphertext pairs. Hence, the proposed frameworks are secure against frequency analysis attacks. □

Lemma 6.0.4. : *The proposed frameworks are secure against man-in-the-middle attacks.*

Proof. The user sends the encrypted data to the CSP. For an attacker to successfully determine the user's actual location, it needs to decrypt the data. As the attacker doesn't have access to the key, it cannot decrypt the data. So it is infeasible to track the user's actual location. Therefore, the proposed frameworks are secure against man-in-the-middle attacks. □

CHAPTER 7

Conclusion

Many researchers have proposed proximity detection schemes based on Bluetooth devices, WiFi co-location, sonar-radar navigation, GPS data, etc. These solutions can detect the proximity but fail to preserve the user's data privacy. We have proposed two privacy-preserving proximity detection frameworks called P3-HD and P3-GH. Both the frameworks use GPS data, restricting the chance of privacy leakage. Without revealing the user's actual locations, it can detect the proximity and alert the user without compromising data privacy. The proposed SADHE detects anomalies while preserving the user's privacy. Without revealing the user's actual location, it identifies the anomalies and removes them to clean the data. The proposed frameworks preserve data integrity, confidentiality, and authentication; only authorized entities have access to the data. We thoroughly evaluated our frameworks concerning privacy and security against potential attacks, including replay attacks, poison attacks, frequency analysis attacks, and man-in-the-middle attacks. We encourage researchers working on proximity detection to search for the best realistic operation point.

References

- [1] E. Barucija, A. Mujcinovic, B. Muhovic, E. Zunic, and D. Donko. Data-driven approach for anomaly detection of real gps trajectory data. In *2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT)*, pages 1–6. IEEE, 2019.
- [2] I. Carreras, A. Matic, P. Saar, and V. Osmani. Comm2sense: Detecting proximity through smartphones. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 253–258. IEEE, 2012.
- [3] C. Cattuto, W. Van den Broeck, A. Barrat, V. Colizza, J.-F. Pinton, and A. Vespignani. Dynamics of person-to-person interactions from distributed rfid sensor networks. *PloS one*, 5(7):e11596, 2010.
- [4] T. Choudhury and A. Pentland. Sensing and modeling human networks using the sociometer. In *Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings.*, pages 216–222. IEEE, 2003.
- [5] M. Dmitrienko, A. Singh, P. Erichsen, and R. Raskar. Proximity inference with wifi-colocation during the covid-19 pandemic. *arXiv preprint arXiv:2009.12699*, 2020.
- [6] J. Krumm and K. Hinckley. The nearme wireless proximity server. In *International Conference on Ubiquitous Computing*, pages 283–300. Springer, 2004.
- [7] D. J. Leith and S. Farrell. Coronavirus contact tracing: Evaluating the potential of using bluetooth received signal strength for proximity detection. *ACM SIGCOMM Computer Communication Review*, 50(4):66–74, 2020.
- [8] E. D. Marks and J. Teizer. Method for testing proximity detection and alert technology for safe construction equipment operation. *Construction Management and Economics*, 31(6):636–646, 2013.

- [9] A. Nieto, S. Miller, and R. Miller. Gps proximity warning system for at-rest large mobile equipment. *International Journal of Surface Mining, Reclamation and Environment*, 19(1):75–84, 2005.
- [10] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.
- [11] V. Patil, P. Singh, S. Parikh, and P. K. Atrey. Geosclean: Secure cleaning of gps trajectory data using anomaly detection. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 166–169. IEEE, 2018.
- [12] G. P. Picco, D. Molteni, A. L. Murphy, F. Ossi, F. Cagnacci, M. Corrà, and S. Nicoloso. Geo-referenced proximity detection of wildlife with wildscope: Design and characterization. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks, IPSN '15*, page 238–249, New York, NY, USA, 2015. Association for Computing Machinery.
- [13] A. Sari et al. A review of anomaly detection systems in cloud networks and survey of cloud security measures in cloud storage applications. *Journal of Information Security*, 6(02):142, 2015.
- [14] Wikipedia contributors. Geohash — Wikipedia, the free encyclopedia, 2021. [Online; accessed 20-April-2022].
- [15] Y. Zheng, H. Fu, X. Xie, W.-Y. Ma, and Q. Li. *Geolife GPS trajectory dataset - User Guide*, geolife gps trajectories 1.1 edition, July 2011. Geolife GPS trajectories 1.1.