Doctoral Thesis

Privacy-preserving Geofencing

Sebastian Zickau

SNET SERVICE-CENTRIC NETWORKING
Technische Universität Berlin
T LABS

Privacy-preserving Geofencing

vorgelegt von
Dipl.-Inform.
Sebastian Zickau
ORCID: 0000-0002-2293-5101

an der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss
Vorsitzender: Prof. Dr. Sebastian Möller
Gutachter: Prof. Dr. Axel Küpper
Gutachter: Prof. Dr. Matthias S. Möller
Gutachter: Prof. Dr. Christian Becker
Tag der wissenschaftlichen Aussprache: 06. Dezember 2022

Berlin 2023

We must defend our own privacy if we expect to have any. We must come together and create systems which allow anonymous transactions to take place. People have been defending their own privacy for centuries with whispers, darkness, envelopes, closed doors, secret handshakes, and couriers. The technologies of the past did not allow for strong privacy, but electronic technologies do.

Eric Hughes, A Cypherpunk's Manifesto (1993)[1]

Arguing that you don't care about the right to privacy because you have nothing to hide is no different than saying you don't care about free speech because you have nothing to say.

Edward Snowden (2015)[2]

Privacy provides the change of pace that makes life worth savoring.

Alan F. Westin, Privacy and Freedom (1967)

The cartographer is the instrument of power as the author of that order, reducing reality to only two conditions: the map and oblivion. The cartographer's truth crystallizes the message that Google and all surveillance capitalists must impress upon all humans: if you are not on our map, you do not exist.

Shoshana Zuboff, The Age of Surveillance Capitalism (2019)

Roads? Where we're going, we don't need roads.

Dr. Emmett Brown, Back to the Future (1985)

A revolution needs more than sensors and data.

unknown

[1] https://www.activism.net/cypherpunk/manifesto.html
[2] https://www.theguardian.com/us-news/video/2015/may/22/edward-snowden-rights-to-privacy-video

Acknowledgements

Throughout the writing of this dissertation, I have received a great deal of support and assistance.

I thank my supervisor, Prof. Dr. Axel Küpper, for his valuable support and guidance during my research projects and dissertation process. I also thank my supervisors, Prof. Dr. Matthias S. Möller, and Prof. Dr. Christian Becker, for their constructive feedback and guidance in the final steps of the writing process. Extra thanks go to Prof. Dr. Sebastian Möller for being the chairman of the doctoral committee.

I thank the Service-centric Networking chair, the Technische Universität Berlin, and the Telekom Innovation Laboratories for giving me the opportunity to become a doctoral candidate. Special thanks to the funding bodies, the German Federal Ministries of Education and Research, their Software Campus program, the German Ministry for Economic Affairs and Technology, and the European Institute of Technology. Their financial support made these results possible.

I thank my co-workers and research students at the SNET chair for their support and valuable input. Special appreciations go to Philip, Dirk, Sebastian, Peter, Sandro, Bersant, Iwailo, Andrea, Sandra, Nicole, Abdulbaki, Konstantin, Katerina, Friedhelm, Martin, Kai, Felix, Marcel, Artjom, Thomas, Jasir, Senan, Hakan, Tanja, María, Tobias, Hai, Boris, Zoltán, Mathias, Gökhan, Ana, Bianca, Johannes, Mohamed, Ilke, and Austin.

Additionally, I also thank all colleagues at the Telekom Innovation Laboratories for coaching, giving industry insides, and providing an exceptional working environment – including fabulous coffee. Individual thanks go to Heinrich, John, Alexander, Irene, Cem, Riccardo, Ulrike, Frank, Monika, Jörg, Anja, Marlene, Roland, Jens, Sven, Daniela, Alexandra, Axel, Joachim, Claudia, Tobias, Maren, Aleksandra, Susanne, and Roxana.

I want to thank the Software Campus colleagues Susanne, Erik, Maren, and Kerstin for their support during the Curcuma project. I thank my project, bachelor, and master students. Huge thanks to Gudrun and Denis for their priceless efforts and feedback during the lengthy writing-up process. I also want to show appreciation to all participants, project partners, and colleagues of the PARADISE project, especially Jonas, who inspired me to work on these particular privacy topics. I am also very grateful to my current Stadt Köln colleagues, Ralf, Thomas, Daryna, and Markus, who gave me the space and time I needed in the final stages of this effort.

In addition, I could not have completed this dissertation without the support of my friends, Steffen, Sybille, Andrea, Anke, Andrea, Stefan, Martin, Gwen, Philipp, Emil, Heike, Hendrik, Annika, Andrea, Ulrich, Peter, and all of the Kleine Spieler:innen and AFOL group. They provided stimulating discussions as well as happy distractions to rest my mind outside of my research. Thank you all.

Finally, I would like to thank my parents, Brigitte and Klaus-Peter, and my brother, Tobias, for their wise counsel and sympathetic ear. I especially would like to thank my girlfriend, Nicole, for her kind support and her imperturbable patience with me. Thank you for always being there for me.

Sebastian Zickau, July 2023

Abstract

Mobile applications that use reactive and proactive location-based services evaluate location information from target devices and users. Services that offer these applications collect user-specific data. Providers tend to use the most accurate location information they can acquire to add value to their services. This information can be sensitive if the target user is in locations that are considered private.

There have been incidents where service providers published data of their users that could be directly linked to individuals and revealed secret locations of areas that were supposed to be kept confidential.

Territory information of different types and semantics can be expressed in the digital world by so-called geofences, i.e., virtual boundaries around specific places and locations. Service providers evaluate precise position information against these geofences and notify the target device depending on the evaluation result.

Based on the evaluation of various geofencing services, this dissertation addresses the creation, representation, and precise evaluation of positions against geofence areas. The focus is on constructing arbitrarily shaped geofences with high accuracy to use exact position information for evaluation.

Novel approaches to privacy-preserving geofencing are presented together with aspects of location-based access control. These approaches include state-of-the-art mechanisms such as zero-knowledge proofs, homomorphic encryption, multi-party computations, attribute-based encryption, cryptographic hashing, etc.

In the evaluation, these approaches are matched with the different requirements based on various use cases. Starting from location-based access control as a security use case, the dissertation focuses on privacy-preserving location-based services. The topic aims to provide a toolbox approach for service providers not relying on trusted third parties to increase their users' privacy and comply with the latest regulations.

Zusammenfassung

Der Privatsphärenschutz von persönlichen Informationen gegenüber Dienstbetreibern ist neben dem Absichern von modernen Client-Server-Anwendungen immer noch eine große Herausforderung für Softwareentwickler. Das Ableiten von Nutzerinnen-Informationen durch direkte und indirekte Datenanalysen ist weit verbreitet. Bei der Nutzung von mobilen Endgeräten fallen Informationen über den Standort der Nutzerinnen an. Dienste nutzen diese ortsbezogenen Informationen u.a. direkt zum Anbieten ihrer Angebote. Indirekt werden durch ungenügende Absicherungen bzw. fehlerhaften oder nicht vorhandene Datenschutzeinstellungen, z.B. des Betriebssystemanbieters, auch private Informationen geteilt. Dabei werden oft unnötig präzise Standortinformationen der Nutzerinnen an Dienste übertragen, ohne dass die Erhebung und Auswertung zwingend notwendig sind.

Die vorliegende Dissertation leitet Methoden und Protokolle her, wie ortsbezogene Dienste die Standortinformationen kodieren können, um eine privatheitsbewahrende Auswertung durch kryptografische Verfahren zu ermöglichen. Hierbei wird das Konzept des *Geofencing* eingesetzt. Beim Geofencing werden virtuelle Grenzen von verschiedenen Gebieten definiert, zwischen denen die mobilen Endgeräte wechseln. Das Wissen, ob ein Endgerät in einem Gebiet ist oder nicht, wirkt sich dann auf die Funktion der Anwendung aus. Bei diesen Anwendungsgebieten ist das Wissen eines präzisen Standorts nicht notwendig, allein das Wissen, ob ein mobiles Gerät in einem Bereich ist oder nicht, reicht zur Ausführung der Anwendungsfunktion aus.

Die vorliegende Arbeit leitet aus verschiedenen Anwendungsszenarien die Ziele der Lösungen ab. Daraus ergeben sich unterschiedliche Anforderungen an das Setup der jeweiligen Methoden und Protokolle. Es werden insgesamt zehn verschiedene Ansätze detailliert hergeleitet, beschrieben und miteinander verglichen. Für die Nutzung dieser kryptografischen Ansätze wird auch eine Punkt-in-Polygon-Auswertung für Geokodierungen eingeführt. Diese Methode ermöglicht es arbiträr geformte Geofences zu erstellen und die genaue Position der Nutzerin dazu auszuwerten.

Die Arbeit wertet Methoden und Protokolle aus Sicht der Anwendungsziele aus und zeigt, dass diese mit aktueller Hardware umsetzbar sind. Zu den kryptografischen Methoden gehören etablierte, basierend auf kryptografischen Hashfunktionen, als auch aktuelle Verfahren, wie z.B. Zero-Knowledge Proofs, kryptografische Akkumulatoren und attributbasierter Verschlüsselung. Diese Ansätze machen eine notwendige vertrauensvolle dritte Partei obsolet.

Als Ergänzung zur Punkt-in-Polygon-Auswertung werden auch inverse Geofences eingeführt. Da die kryptografischen Schemata größtenteils auf Element-in-Menge-Verfahren bestehen, kann von der Nutzung von einer Breiten- und Längengrad-Auswertung ab-

strahiert werde, um die Ansätze auch bei anderen Lokalisierungsverfahren, z.B. mittels Bluetooth- oder WLAN-IDs, nutzen zu können. Zusätzlich können die Geokodierungen auch für Anwendungen genutzt werden, bei denen der Abstand zweier Nutzerinnen Privatsphären-bewahrend geprüft werden soll, wie z.B. bei Kontaktnachverfolgungsapps zur Eindämmung von Pandemien.

Notation

$\mathbb{N}$	natural numbers
$\mathbb{Z}$	integers
$\mathbb{Q}$	rational numbers
$\mathbb{R}$	real numbers
$a \mid b$	$a \in \mathbb{Z}$ divides $b \in \mathbb{Z}$
$\lvert x \rvert$	absolute value of $x \in \mathbb{R}$
$\lvert x \rvert$	length of a bit string x
$\lvert x \rvert$	binary length of $x \in \mathbb{N}$
$\lvert M \rvert$	number of elements in a set M
M^*	set of words $m_1 m_2 ... m_l, l \geq 0$, over M
$\{0,1\}^*$	set of bit strings of arbitrary length
1^k	constant bit string 11...1 of length k
$a \oplus b$	bitwise XOR of bit strings $a, b \in \{0,1\}^k$
$a \lvert\lvert b$	concatenation of strings a and b
$\vec{a}$	vector $\vec{a}$
$\vec{a} \cdot \vec{b}$	dot product of vectors $\vec{a}$ and $\vec{b}$
$g \circ f$	composition of maps: $g \circ f(x) = g(f(x))$
id_X	identity map: $idX(x) = x$ for all $x \in X$
f^{-1}	inverse of a bijective map f
x^{-1}	inverse of a unit x in a ring
$\mathbb{Z}_n$	residue class ring modulo n
$\mathbb{Z}_n^*$	units in $\mathbb{Z}_n$
a div n	integer quotient of a and n
$a \bmod n$	remainder of a modulo n
$a \equiv b \bmod n$	a congruent b modulo n
$gcd(a,b)$	greatest common divisor of integers
$lcm(a,b)$	least common multiple of integers
$\mathbb{F}_q$, $\mathrm{GF}(q)$	finite field with q elements
$E(\mathbb{F}_q)$	$\mathbb{F}_q$-rational points of an elliptic curve
$ord(x)$	order of an element x in a group
QR_n	quadratic residues modulo n
QNR_n	quadratic non-residues modulo n
$[a,b]$	interval $a \leq x \leq b$ in $\mathbb{R}$
$\lfloor x \rfloor$	greatest integer $\leq x$
$\lceil x \rceil$	smallest integer $\geq x$
$O(n)$	Big-O notation
Primes_k	set of primes of binary length k
P or $P(X)$	positive polynomial
$E(R)$	expected value of a random variable R

$prob(\mathcal{E})$	probability of an event $\mathcal{E}$
$prob(x)$	probability of an element $x \in X$
$prob(\mathcal{E}, \mathcal{F})$	probability of $\mathcal{E}$ AND $\mathcal{F}$
$prob(\mathcal{E}\|\mathcal{F})$	conditional probability of $\mathcal{E}$ assuming $\mathcal{F}$
$prob(y\|x)$	conditional probability of y assuming x
$X \bowtie W$	join of a set X with $W = (W_x)_{x \in X}$
XW	joint probability space
$x \xleftarrow{p_X} X$	x randomly selected according to p_X
$x \xleftarrow{R} X; x \in_R X$	x randomly selected from X
$x \xleftarrow{u} X$	x uniformly selected from X
$x \xleftarrow{R} X, y \xleftarrow{R} Y_x$	first x, then y randomly selected
$prob(\ldots : x \leftarrow X)$	probability of ... for randomly chosen x
$x \leftarrow S$	x randomly generated by the random variable S, seed
$y \leftarrow A(x)$	y randomly generated by A on input x
$dist(p, \tilde{p})$	statistical distance between distributions
$H(X)$	uncertainty (or entropy) of X
$H(X\|Y)$	conditional uncertainty (entropy)
$I(X;Y)$	mutual information
$\Pi(S)$	permute order of elements in Set S

Contents

We'll always have Paris.

Casablanca (1942)

1 Introduction

> *... My general conclusions are that perhaps in 30 years the orbital relay system may take over all the functions of existing surface networks and provide others quite impossible today. For example, the three stations in the 24-hour orbit could provide not only an interference and censorship-free global TV service for the same power as a single modern transmitter, but could also make possible a position-finding grid whereby anyone on earth could locate himself by means of a couple of dials on an instrument about the size of a watch. (A development of Decca and transistorisation.) It might even make possible world-wide person-to-person radio with automatic dialling. Thus no-one on the planet need ever get lost or become out of touch with the community, unless he wanted to be. I'm still thinking about the social consequences of this! But as for details of frequencies and powers, I'll have to leave that to the experts to work out; I'll get on with my science fiction and wait to say 'I told you so!'*
>
> Arthur C. Clarke (1917 – 2008, author of 2001: A Space Odyssey),
> in a letter to Andrew G. Haley (1956)

More than 70 years ago, in 1956, author Arthur C. Clarke imagined a world-wide positioning system and communication framework that can be viewed as an initiator for today's well-known Global Positioning System (GPS), originally named Navstar GPS (Navigational Satellite Timing and Ranging GPS), which started as a military service in 1973 [1]. Its prototype spacecraft was sent into orbit in 1978. Its entire constellation was usable much later – in 1993. Its civil use was introduced in the year 2000. Most notable is the increased accuracy of the system also when used combined with other global navigation satellite systems (GNSS).

The letter also mentioned a world-wide stationary independent communication system. Along with GPS, wireless communication standards emerged from 1G in the 1980s to the latest well-known 4G and 5G standards. The advent of smartphones combined GNSS and Internet communication with social and messaging applications. Clarke not only *predicted* this development but was also worried about its *social consequences*.

1.1 Mobile applications

Through the introduction of the first iPhone in 2007, several other mobile devices became an integrated part of our daily life. These devices are laptops, netbooks, tablets, and smartwatches/fitness trackers, to name a few. Even small computers, such as the Raspberry Pi, can be considered mobile devices, as they are small and light enough to move around. This device can also be regarded as an Internet of Things (IoT) device that got more popular in recent years. Additional to the relatively low-cost consumer devices, there are also professional devices, such as medical apparatuses.

This means that the digital age of processing data on-the-go is a reality today, enabling people to work wherever they choose. However, this freedom of whereabouts comes with new possibilities to include location information in the digital process. It is common to use the user's current location information or the mobile device's location, i.e., the *target client*, to be sent to the requested service or within a mobile application. Examples of such usages are finding the closest carsharing vehicle or sharing your location with friends to meet them.

There are different ways to represent location information digitally. Common representations are GPS coordinates and addresses. One can convert an address to coordinates using a process called *geocoding* and *reverse geocoding* for its inverted process. The GPS coordinate format is standardized and used the same way around the globe. Details are presented in the related work chapter. Addresses, however, can be different for each country or vary in different cultures. Sometimes there are numbered postal codes, letter-based ones, or even none used at all, i.e., in Ireland.

Additionally, technology and techniques can obtain location information, such as Cell-IDs, WLAN-IDs, and Bluetooth-IDs. They also use different formats, such as iBeacon or Eddystone, in Bluetooth Low Energy (BLE). Integrating those IDs into a service or application usually requires databases matching those IDs to addresses or coordinates. New technologies are already emerging, meaning that location information representation options are likely to increase in the near future.

As with the GPS coordinates, there is a general understanding of the data representation and the common knowledge that particular longitude-latitude values, 2°17´40.2´´E 48°51´30.9´´N, represent a location east of the prime meridian in Greenwich in England and on the northern hemisphere. This information would lead to the point being located at the Eiffel Tower in Paris, France. A computer would instead use the coordinates in their decimal representation, i.e., 48.858575, 2.294492.

These are examples of using common address formats and GPS coordinates. GPS is the US-American system of different global navigation satellite systems, such as GLONASS (Russian), Galileo (European), and BeiDou-2 (Chinese). GPS started as a military project and was made available to the public. GPS coordinates are usually represented with the latest revision of the World Geodetic System (WGS 84) [2].

A device's location is unique if its representation is very precise, e.g., in the case of WGS 84, such as 40°41´21.3´´N 74°02´40.3´´W. In the case of a city or country name, it is not very precise, e.g., Berlin or Spain, as there are different places called Berlin. Mobile devices can use exact locations, most accurate for an outdoor scenario. The device needs to receive direct signals from the satellites, which can be difficult for indoor areas. The

precision also depends on additional factors, such as weather conditions, large buildings, or other obstacles close to the target client.

The location can be seen as a verifiable characteristic of a mobile device. As mentioned above, these verifiable characteristics can be included in a service call or an application. It will alter the result accordingly. Verifiable credentials are also used within *security* applications, such as passwords, tokens, keys, and biometrical values. The security aspect is expressed in the term location-based access control (LBAC). According to an access policy, a device's location is used to grant or deny access to a file, a service, or an application. Within the *privacy* use cases, location information and its interpretation are used to obscure a device or user location.

One year after the European General Data Protection Regulation (GDPR) had come into effect, many companies tried to address the privacy concerns of their users with new application settings [3] and limitations on API levels for services [4]. However, these were reactions to location-based privacy incidents. Some of the most talked-about services include Strava [5], Uber [6], e-scooter tracking [7], and the dating service Loovo [8]. Recently, an app developer SDK from UK-based company Huq Industries made headlines [9] [10].

Analog to implementing security features within an app or service, avoiding privacy traps is equally challenging. Additionally, knowing, processing, and using user data is also a viable business opportunity for companies, as thoroughly summarized by Shoshana Zuboff in her 2019 book *The Age of Surveillance Capitalism* [11]. That means a company needs to put more effort into developing *privacy-preserving* services but loses possible business opportunities.

This thesis presents real-world use cases and services from which the users would benefit and provides an overview of state-of-the-art methods for implementing such privacy-preserving services while retaining their functionalities *and* complying with the law.

The privacy aspects are the thesis' focus and are motivated by use cases from different research projects. The domains include vehicle rental, anti-doping control, and citizen services. The different areas will ensure that the proposed system is usable in a variety of settings that are motivated by use case-specific requirements, see Section 2.4.

Different approaches are considered, adapted, detailed, and evaluated. The following points are addressed within this thesis.

- Location representation and encodings
- Verification and evaluation of point-in-polygon information
- Privacy-preserving schemes and protocols based on cryptographic primitives
- Incorporation of these into additional use cases and application scenarios

The thesis statement, research questions, contribution, and out-of-scope topics are detailed in Section 2.7.

1.2 Privacy

Regarding whereabouts, private location information to carry out additional services has been a topic in recent news. Facebook is experimenting with using location information to suggest new friends, even when the location is considered personal, e.g., self-help

groups [12]. The popular location-based game Pokémon Go was used in unappropriated areas, such as holocaust memorials and cemeteries [13]. These are only some examples of conflicting privacy interests between services and whereabouts.

Privacy-by-Design (PbD) and Privacy Enhancing Technologies (PETs) are two popular approaches covering various techniques and technologies to increase user privacy in the modern age of internet communication. With the advent of mobile devices and the implications of location-based services (LBS), the idea of privacy-preserving functionalities quickly became a topic within the research community. The usage of different positioning methods increased over the past years, including precise indoor positioning techniques and an immanent positioning of mobile devices using Wi-Fi access points (APs), Cell-IDs, and assisted GPS (A-GPS).

The privacy model suggested in this work bases its requirements on a particular use case scenario. In this scenario, a single user *A* needs to be located as precisely as possible whenever a single requester *D* has the assignment to identify the whereabouts of user *A*. This must be done without informing user *A* about this action. In our use case, user *A* either uses a modern smartphone or a dedicated device *E*, such as a small wearable, which can be re-actively pinged via a backend service. We call this backend service the LBS provider. The device *E* uses an external location provider to calculate its current position or uses an internal database to calculate its position. In both cases, device *E* sends this information back to the LBS provider.

In this delicate use case, we consider both the service provider and the location provider as untrusted parties. In this work, different implications are discussed, and what parts of the architecture can be trusted. Mechanisms are introduced to grant privacy to the user even if different parts of the players are not trusted. The goal is to design a system that allows user D to request user A's position on a 24/7 basis and grants him a maximum level of self-determined privacy. The use case also lies under the current valid European General Data Protection Regulation (EU-GDPR).

This thesis focuses on applying practical cryptographic methods to enable privacy-preserving geofencing with arbitrarily shaped (geo)fences. In Chapter 2, the topic will be motivated by real-world use cases.

A dictionary definition of the term *privacy* is given below.

privacy | ˈprīvəsē |
noun
The state or condition of being free from being observed or disturbed by other people: she returned to the privacy of her own home.

Examples of private places Additionally, the Strava data revealed that there are private places one does not want to be disclosed in relation to one's personal identity information. This could have negative consequences on someone's private or professional life. Several example places to be considered confidential, such as religious places, counseling sessions, alcoholics anonymous meetings, prisons, cemeteries, and brothels, are visualized in Figure 1.1.

[1] Image source: Icons used throughout the thesis representing locations, users, and roles are from a subscription to https://thenounproject.com/.

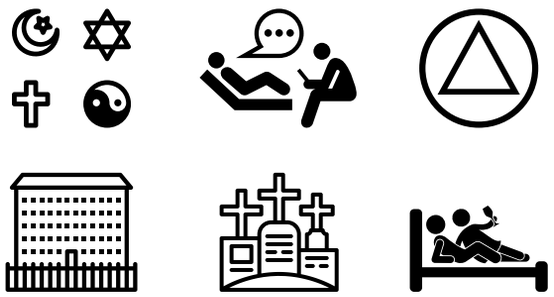


Figure 1.1: Location and place examples that are considered very private for people.[1]

It has to be remembered that the revelation of the Strava data was no breach done by a hacker. However, the information was offered *carelessly* in the act of openness to advertise the service.

1.3 Geofence and geofencing

The following general explanation for *geofences* and *geofencing* is taken from [14]. A scientific definition is given in Chapter 3.

> *A geo-fence is a virtual perimeter for a real-world geographic area. A geo-fence could be dynamically generated—as in a radius around a store or point location. Or a geo-fence can be a predefined set of boundaries, like school attendance zones or neighborhood boundaries. Custom-digitized geofences are also in use. When the location-aware device of a location-based service user enters or exits a geo-fence, the device receives a generated notification. This notification might contain information about the location of the device. [...] Geofencing, used with child location services, can notify parents when a child leaves a designated area. [...] It allows users of the system to draw zones around places of work, customers sites and secure areas. These geo-fences when crossed by an equipped vehicle or person can trigger a warning to the user or operator via SMS or Email. [...] Geofencing in a security strategy model provides security to wireless local area networks. This is done by using predefined borders, e.g., an office space with borders established by positioning technology attached to a specially programmed server. The office space becomes an authorized location for designated users and wireless mobile devices.*

The spelling without a hyphen is used throughout this thesis, unlike the spelling in some quoted texts.

Geofence interfaces Figures 1.2 and 1.3 show different geofencing interfaces. One shows the iOS interface for location-based reminders. A circular geofence can be created, and a reminder can be set when *leaving* or *entering* this area.

Figure 1.3 shows the Strava interface for setting privacy zones [15]. This interface can also be used to draw circular geofences around addresses/locations with a variable radius. This feature was introduced by Strava to only upload personal GPS data to the service

[2]Image source: Screenshots of the Apple iOS Reminders app.

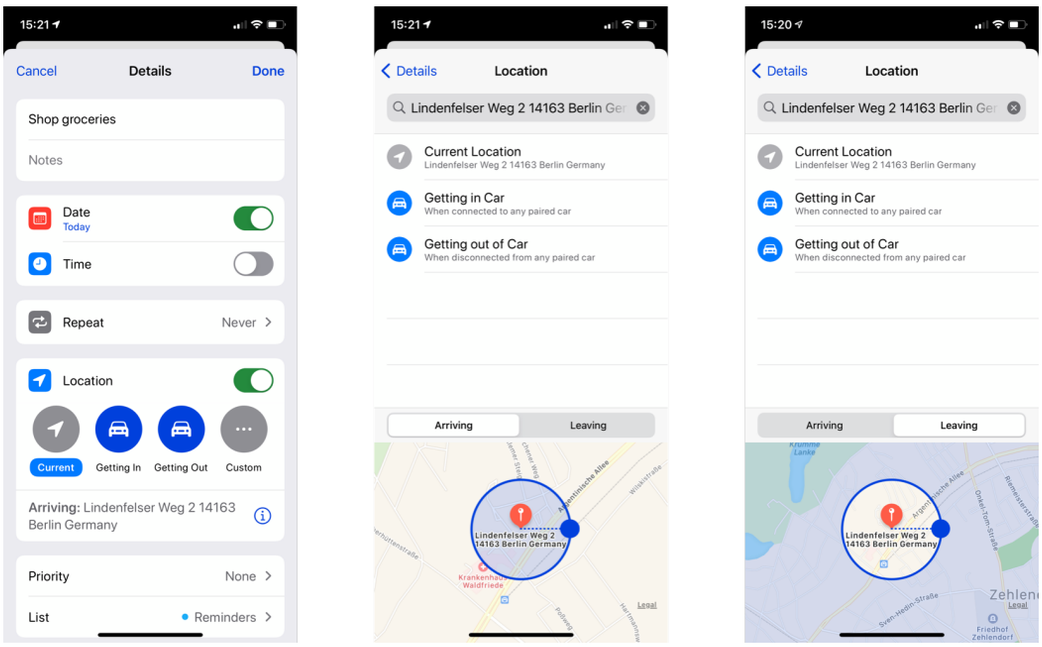


Figure 1.2: Interface for iPhone reminders with geofences.[2]

outside the radius to preserve the privacy of a user's address. However, studies have shown that these simplified privacy features are limited in their effectiveness [5] [16].

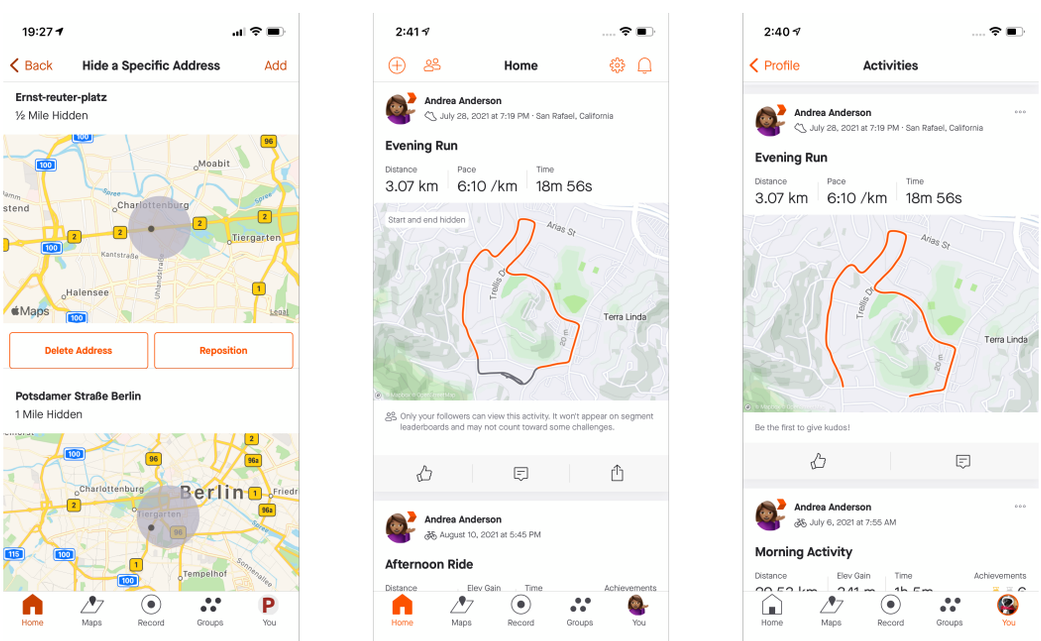


Figure 1.3: Strava interface for privacy zones using geofences.[3]

1.4 The *Where is Waldo?* analogy

The *Where is Waldo?* pictures and books, also known as *wimmelbooks*, *hidden picture books* or *Wimmelbilder* (German), are based on the idea showing depicting scenes richly

[3]Image source: Screenshots of the iOS Strava app and examples from https://support.strava.com/.

detailed with humans, animals, and objects, e.g., a person is hidden in a crowd of hundreds of people. Figure 1.4 shows an early example, the painting *Netherlandish Proverbs* (1559) by Pieter Brueghel the Elder [17], which presents 100 Dutch proverbs in a village near the sea stage. The task is to skim through the picture to locate Waldo, who mostly wears red and white clothes and sometimes waves his hand.

Figure 1.4: Pieter Bruegel the Elder, Netherlandish Proverbs. 1559, an early example of a hidden object painting.[4]

Figure 1.5 shows a Waldo example [18] in a scene representing a bustling train station. Usually, the pictures represent an everyday situation or thematic and historical sights. The books are primarily addressed to children to impart persistence and concentration. With this goal in mind, it is evident that Waldo can be found in these pictures.

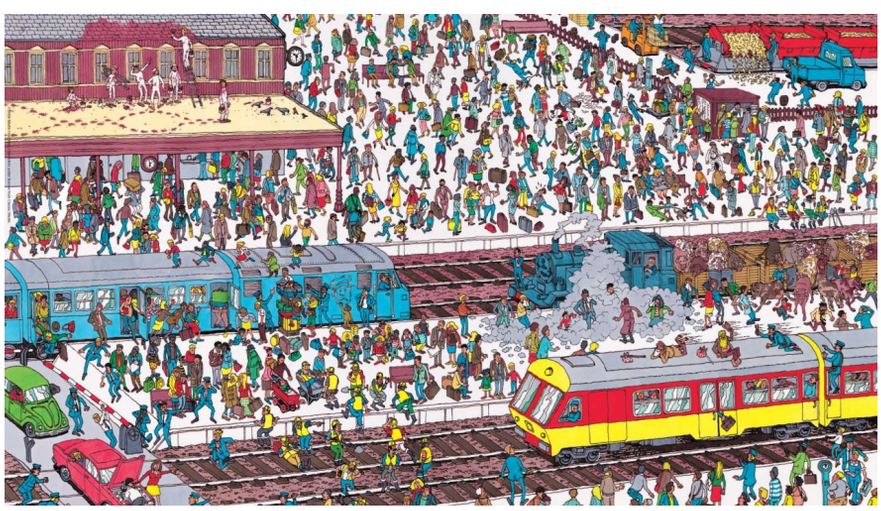


Figure 1.5: A Where is Waldo? example picture.[5]

[4]Image source: https://www.wga.hu/frames-e.html?/html/b/bruegel/pieter_e/03/01prover.html
[5]Image source: Example image is from the 2017 book Where's Wally? The Totally Essential Travel Collection by

The analogies that this thesis address is based on three relations to the *Where is Waldo?* pictures.

1. How can person A prove to person B that person A knows where Waldo is *without* revealing his location in the picture by pointing at him?
2. In the presented cases of the thesis, the variation is that Waldo can also be *not* present in the picture.
3. Let the *edge* of the picture be an arbitrarily shaped geofence. Waldo is inside the geofence or not, see Figure 1.7.

Figure 1.6: A simple rectangular geofence representing, for example, the area of the Waldo image.

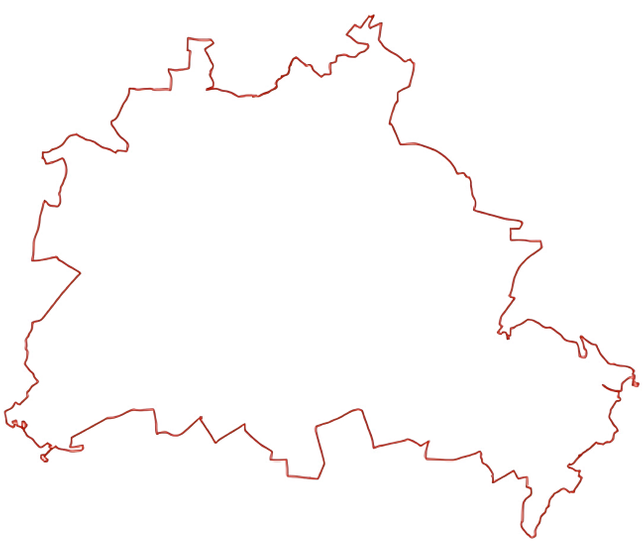


Figure 1.7: An arbitrarily shaped geofence representing, for example, a city boundary.

Figure 1.6 shows a simple geofence, i.e., the same 16 by 9 ratio that the Waldo picture shows. In Figure 1.7, this geofence has the arbitrary shape of Berlin, Germany.

So the game of *Where is Waldo?* will become the thesis' focus of *Is Waldo in the picture or not?* – without revealing his location!

Martin Handford, Walker Books Ltd, digital image is used from https://www.spacetimestudios.com/attachment.php?attachmentid=16040&d=1346543064.

1.5 Thesis outline

The thesis' primary focus is on applying cryptographic primitives, schemes, and protocols for a privacy-preserving evaluation of precise location data. The research on this topic was preceded by applying security methods to location-based access control use cases. This research is partly presented as a motivation and inspiration. This thesis is structured as follows.

Chapter 2 motivates the research carried out for the presented use cases. The chapter starts by explaining the legal framework. It also covers general privacy approaches. The chapter derives 17 generalized objectives from seven use cases. It concludes with a brief overview of the research methodology.

Chapter 3 is reserved for defining aspects of location-based services. It presents the active roles and actors and the differences in security and privacy models. Chapter 3 closes with an overview of possible attacks and their countermeasures.

Chapter 4 gives an overview of geo encoding systems. These encodings also form the basis for arbitrarily shaped geofences and precise position information. Chapter 4 presents previous methods and protocols that enable privacy-preserving location-based applications. The chapter also cites a taxonomy of location-based privacy methods, which will be extended as part of the contribution. Section 4.4 presents the mathematical and cryptographic schemes for the thesis' primary approaches as an addition to the definitions in Chapter 3 and as a foundation for Chapter 5.

Chapter 5 will present the thesis' main contributions on how cryptographic primitives, schemes, and protocols are used to define protocols to address the use cases and their 17 derived objectives presented as part of the motivation. The chapter builds on the geo encoding systems and cryptographic primitives to use location data in a privacy-preserving geofence service. Additional use cases, such as area limitation proofs, distance measuring, and contact tracing, are also discussed, and how the overall concepts can be integrated into existing infrastructures.

In Chapter 6, proof of concept implementation details will be given for various scenarios.

Chapter 7 extensively evaluates the schemes and protocols, focusing on computational feasibility. Additional results are given on how these schemes address privacy attacks.

Finally, Chapter 8 will conclude the thesis by assessing how the privacy methods would be used by location-based services and an overview of their current limitations. The thesis closes with an outlook on future research topics and a solution for the practical zero-knowledge proof of the *Where is Waldo?* analogy.

Furthermore, the appendices present cryptographic primitives, schemes, and protocols in more detail, an overview of the author's publications, research projects, and supervised theses. The lists of figures, tables, listings, and the bibliography are at the very end of the document.

1.6 Chapter summary

The chapter introduced the areas of the thesis, mainly why privacy matters in location-based services. The notion of geofences and geofencing was presented. Additionally, a

practical analogy was given in the form of a *Where is Waldo?* picture and the question of how to solve it in a zero-knowledge fashion. An answer on how to do this is given at the end of the conclusion chapter. Additionally, the thesis structure was also outlined.

Houston, we have a problem.

Apollo 13 (1995)

2 Motivation

In the book *Mobile Positioning and Tracking* [19], Figueiras et al. briefly describe the relationship and their interdependency of the terms *positioning*, *location*, *time*, and *tracking* as follows.

> *Positioning or location can be understood as the unambiguous placement of a certain individual or object with respect to a known reference point. [...] Although the position itself is obviously a significant source of information related to a specific time, it is even more useful. In particular, when we consider tracking systems, time information is a key necessity, not only for knowing the position of a certain device at a specific time but also for inferring higher-order derivatives of the position, that is, speed and acceleration. Thus, "tracking" is a method for estimating, as a function of time, the current position of a specific target.*

In the use cases described in detail in Section 2.4, the general setting is that an individual is consuming a service using a positioning device. In most cases, the device is a smartphone capable of positioning itself in relation to the Earth's coordinate system and sending this encoded position via a network to the service provider to consume a functionality, e.g., a navigation system.

As introduced in Chapter 1, there are various examples in which a third party in the form of an application service provider gathers information, willingly or unwillingly, about their users, including the device location. This location data becomes privacy-sensitive if the service provider can infer information and knowledge about its users from additional user data, such as name, age, address, profession, and gender, together with publicly available data about medical institutions' addresses, counselors, self-help groups, or religious places. The service may also infer information and knowledge from combining additional data from other users or evaluating tracking paths and time intervals. The mentioned roles and actors are defined in Section 3.2.

2.1 Definitions of Privacy

Alan Westin defines his understanding of *privacy* in his book *Privacy and Freedom* [20] from 1967.

> *Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others. Viewed in terms of the relation of the individual to social participation, privacy is the voluntary and temporary withdrawal of a person from the general society through physical or psychological means, either in a state of solitude or small-group intimacy or, when among larger groups, in a condition of anonymity or reserve.*

Westin describes four primary states of individual privacy, *solitude, intimacy, anonymity*, and *reserve*. His take on the state of *anonymity* reads as follows.

> *The third state of privacy, anonymity, occurs when the individual is in public places or performing public acts but still seeks, and finds, freedom from identification and surveillance. He may be riding a subway, attending a ball game, or walking the streets; he is among people and knows that he is being observed; but unless he is a well-known celebrity, he does not expect to be personally identified and held to the full rules of behavior and role that would operate if he were known to those observing him. In this state the individual is able to merge into the "situational landscape." Knowledge or fear that one is under systematic observation in public places destroys the sense of relaxation and freedom that men seek in open spaces and public arenas.*

In his work, Alan Westin addresses individuals' privacy in public places, not only in one's home. The independence of privacy and a human being's location is what he refers to as *to merge into the situational landscape*. This thesis' primary objective is to provide and increase privacy for citizens independent of their location and the LBS services they consume.

2.2 Legal aspects

This section presents the legal regulations' motivational aspects, focusing on the European perspective.

2.2.1 European Convention on Human Rights

The aspect of privacy is addressed in Article 8 of the *Right to respect for private and family life* of the European Convention on Human Rights in 1950 [21].

> *1. Everyone has the right to respect for his private and family life, his home and his correspondence.*
>
> *2. There shall be no interference by a public authority with the exercise of this right except such as is in accordance with the law and is necessary in a democratic society in the interests of national security, public safety or the economic well-being of the country, for the prevention of disorder or crime, for the protection of health or morals, or for the protection of the rights and*

freedoms of others.

The following quote is cited from the European Court of Human Rights website [22].

> *The Convention for the Protection of Human Rights and Fundamental Freedoms, better known as the European Convention on Human Rights, was opened for signature in Rome on 4 November 1950 and came into force in 1953. It was the first instrument to give effect to certain of the rights stated in the Universal Declaration of Human Rights and make them binding.*

2.2.2 General Data Protection Regulation

As the convention addresses privacy protections for citizens from public authorities, the European General Data Protection Regulation (GDPR) also addresses privacy regulations regarding private organizations and businesses offering (digital) services to European citizens. Nevertheless, the GDPR also applies to government agencies, including police and military, and data stored within the EU [23]. The regulation came into force on May 25th, 2018. It repeals its predecessor, the Directive 95/46/EC, from 1995. The GDPR is effective for all nations in the European Economic Area (EEA); these are the 28 member states of the EU, with Iceland, Lichtenstein, and Norway (as of 2020) [24].

2.2.2.1 Key roles and terms

These are the key roles of the GDPR (Article 4) legal text and a short description. A complete role and actor definition for the thesis is given in Section 3.2 [24].

- **Natural person**: A human being
- **Data subject**: A directly or indirectly identified or an identifiable natural person who can be referenced by an identifier, such as a name, a number, or location data
- **Controller**: A natural or legal person, public authority, agency, or other body that decides on the means and purposes of processing personal data
- **Processor**: The entity (natural or legal) processing a subject's data, either on behalf or in agreement with the controller. The controller and processor can be the same entity
- **Third party**: A natural or legal entity that processes personal data under the supervision of a controller or processor

The main objectives of the GDPR addressed in Article 1 are to provide every individual, i.e., the data subject, with the fundamental right to protect personal data and enable maximum control over their data [24]. This thesis aims to develop methods to protect the data subject's whereabouts from the data controller and processor, exemplified by the use cases described in Section 2.4.

Article 4 of the GDPR defines *personal data* as information that can be used to *identify* a natural person, i.e., the data subject. This information includes *location data*. The article defines *processing* as any *operation on personal data* and demands *restrictions* on the data processed. The GDPR addresses the possible analyses and prediction *aspects concerning the natural person's location and movement*. Data about *physical or mental health* is categorized as special (Article 9). This category includes data revealing *racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, a natural person's sex life, or sexual orientation.*

In Article 5, the GDPR defines the principles governing personal data processing. The most relevant principle is *data minimization*, which is described as *adequate, relevant, and limited to what is necessary in relation to the purposes*. The GDPR necessitates data *integrity and confidentiality* by *using appropriate technical or organizational measures*.

Article 22 states that the *data controller shall implement suitable measures to safeguard the data subject's rights and freedoms and legitimate interests*.

2.2.2.2 Data protection measures

Article 25 demands that the controller uses mechanisms to ensure *data protection by design and by default*. This is expressed as *at the time of the processing itself, implement appropriate technical and organizational measures*.

2.3 Privacy approaches

There are several privacy approaches in the literature. Two relevant to this thesis' methods are briefly summarized, *privacy by design* and *privacy engineering*.

2.3.1 Privacy by design

As the GDPR demands protection by design and default, the seven foundational *Privacy by Design* (PbD) principles by A. Cavoukian et al. [25] are presented.

1. Proactive, not reactive; preventive, not remedial
2. Privacy as the default setting
3. Privacy embedded into the design
4. Full functionality – positive-sum, not zero-sum
5. End-to-end security – full lifecycle protection
6. Visibility and transparency – keep it open
7. Respect for user privacy – keep it user-centric

The thesis focuses on the following three aspects based on the definition in [25].

3. Privacy embedded into design

   A service, i.e., a processor, should construct its system's architecture and procedures with privacy in mind to become an inherent part of the overall functionality.

4. Full functionality - positive-sum, not zero-sum

   The goal is to provide a similar service without compromising functionality or trade-offs regarding the data subject's private information.

5. Respect for user privacy - keep it user-centric

   The user's rights and privacy assumptions need to be the focus of service architectures and operations by including strong and state-of-the-art cryptographic measures.

PbD is a modified approach of *Value sensitive design* (VSD) by Batya Friedman et al. [26] and later, together with David G. Hendry [27].

2.3.2 Privacy engineering

Privacy engineering is the field of developing approaches, measures, and architectures to ease the process of developing privacy-preserving applications. The National Institute of Standards and Technology (NIST) definition focuses on increasing privacy and enabling organizations to make intentional decisions about effective and practical implementations [24].

The European Internet Privacy Engineering Network (IPEN) stated in 2015 the following shortcoming [28] in terms of introducing privacy to applications. First and foremost, *the deficit of attention to privacy in development lacks appropriate tools and best practices*. The development processes are rushed, re-using older components despite the lack of privacy mechanisms. Security is often too complex to be thoroughly integrated into the business processes and their software pendants.

In summary, this motivates the applied character of the privacy-preserving approaches and the combination of cryptographic tools presented in this thesis.

2.4 Use cases

The following use cases are addressed in this thesis. The focus is on the privacy-preserving applications, which were developed following the research carried out on location-based access control (LBAC) [29] [30]. The LBAC scenario was motivated within a hospital setting [31] [32] [33] [34]. Private patient data was secured via geolocation access policies utilizing the Geo eXtensible Access Control Markup Language (GeoXACML) [35] Cell-IDs, WLan-IDs, BLE-beacons, and GPS information. GeoXACML is based on the OASIS XACML standard that defines a declarative fine-grained, attribute-based access control policy language using the Extensible Markup Language XML [36]. Additionally, the standard defines an architecture and a processing model that evaluates access requests according to the rules that a user defines in the (Geo)XACML policies.

Following the ideas from the access control policies, the privacy-preserving use cases and methods emerged from a project that aimed at identifying technologies and services to enhance the privacy of professional athletes who need to be available for unannounced doping tests [37]. The focus was on preserving the athlete's privacy in general, particularly regarding their *whereabouts* [38] [39].

From both domains, the notion of *geofences* was inherently used. Regarding security methods, geofences describe where a user needs to access data. In the case of privacy-preserving scenarios, the notion is used to describe areas where users see themselves as being in a place that is considered private. As described in Chapter 1, such sites are connected, for example, to religious, medical, or intimate matters. The definition of geofences and geofencing and their notion in this work is given in Section 3.1.1.

The concepts described in the subsequent chapters enable their usage in additional domains where evaluating single positions against areas that are describable with arbitrarily shaped geofences is practical. An example is the described usage of geofences within distributed ledger technologies (DLT).

Additionally, *privacy-preserving distance measuring* between two or more devices and *speed measuring* are included in the thesis' outcome, expanding the current research.

2.4.1 Athlete's whereabouts

Before the 2016 Olympic Games in Rio de Janeiro, several athletes were banned from participating due to positive doping tests and several sports associations' irregularities. These events brought the topic of doping tests and the procedures for executing them back into the limelight. The Anti-Doping Administration & Management System (ADAMS) [40] is currently used by most of the national anti-doping organizations (NADO). ADAMS uses the whereabouts information of athletes to realize unannounced doping tests.

The work presented in this thesis is based on the privacy use case of doping controls for athletes, namely researched in the project *Privacy-enhancing And Reliable Anti-Doping Integrated Service Environment* (PARADISE) [41]. ADAMS is hosted by the World Anti-Doping Agency (WADA) based in Canada. NADOs, including the German Nationale Anti-Doping Agentur (NADA), use ADAMS to plan and realize worldwide doping controls, including storing results such as athletes' blood and urine samples. They also save information on therapeutic usage exemptions (TUEs), i.e., medication athletes with conditions can take.

An unannounced doping control has to be carried out to test athletes for listed substances on all possible 365 days a year, commonly between 6 am and 10 pm. Exceptions are also possible if pieces of evidence are apparent. The general process involves a NADO, which sets a date and time or a date range for an imminent doping test and informs a subcontractor or the doping control officer (DCO) directly about the specific athlete. Depending on the current athlete's training and competition schedule or previous results, either a blood test or a urine test is carried out.

The athletes provide information about their whereabouts in a calendar-like system every quarter in the current system. They have to state each overnight stay address, including names of partners or hotels and their regular whereabouts, such as work, school, and training facilities. The athletes need to keep this information up-to-date if changes occur. The NADOs and the DCOs use these calendar entries to plan and conduct their doping tests.

The current ADAMS system has several drawbacks from usability, privacy, and security points of view. It needs to be stated that there is no full access to the overall specification of the system. The ADAMS system and the doping control planning processes were conducted with athletes, NADA employees, media coverage, and online resources during the PARADISE project.

Drawbacks of the ADAMS system

The following shortlist describes the main disadvantages of the ADAMS system.

- More whereabouts information is provided than necessary.
- On-the-fly changes cannot easily be made.
- It is not clear who has access to the whereabouts information.
- The visibility of whereabouts information cannot be limited.
- The granularity of the whereabouts to be provided is not clear.
- An athlete may not know if the whereabouts information is deleted after becoming obsolete.
- A DCO seems to have access to more whereabouts information than necessary.
- The system was vulnerable to cybercrime attacks [42].
- It is unclear if information accessed by ADAMS users is logged for transparency

reasons.

The German-funded R&D project PARADISE [41] aimed to address the privacy-related whereabouts challenges.

PARADISE's privacy-preserving approach

The brief description of the current system highlights some privacy issues, such as not knowing who has access to the data or that the athletes need to disclose their whereabouts three months in advance. Also, the system is hosted in a country other than the origin of the athlete.

The data privacy project PARADISE addresses the shortcomings of the current system. Within the project, the main focus was on improving the athletes' privacy aspects and improving the overall usability for the active roles, athletes, DCOs, national anti-doping agencies, and their subcontractors.

The project PARADISE aimed to introduce a dedicated device called *eves* (analog to ADAMS). The device was seen as an addition to the ADMAS system, and its usage was meant to be voluntary. This device was supposed to submit its location to the backend system if a DCO requested it. It has to be emphasized that the device is called reactively and that only the DCO carrying out the control can request the location. The system was designed so that it cannot be used to track athletes. Additional privacy functions were introduced by the notion of *privacy geofences*, called *privacy gardens*, and the idea of using *semantic geofences* to reveal only relevant information. For the latter use case, the position of the DCO was also included while computing the request. This last approach is described as policy-based geofencing in Section 5.3.9.

A general example of such functionality is the dedicated mobile application Glympse [43], which can be used to proactively share one's location with friends for a given period. Today such functionality is also integrated into messaging apps such as WhatsApp.

Doping control and the GDPR

Article 49 of the GDPR declares derogations for specific situations, including transferring personal data to international organizations, if this complies with public interests. This includes cases such as *contact tracing for contagious diseases or [...] to reduce and/or eliminate doping in sport* [23]. The mentioned contact tracing use case is described in Section 2.4.5.

2.4.2 Carsharing

With the advent of the sharing economy (e.g., AirBnB) in recent years, carsharing services emerged [44]. These classical carsharing solutions, such as ShareNow (present in eight European countries) [45] and Miles Mobility (present in eight German cities) [46], are operated by the automotive companies Daimler AG and BMW. The services recently merged under the label ShareNow. The companies own the cars shared. Carsharing in its various forms dates back to the 1940s in Switzerland, but more versatile services arose with the introduction of intertwining technologies, like GPS- and internet-enabled devices.

The main focus is on peer-to-peer (P2P) carsharing as a form of collaborative consumption. In this approach, private-owned cars are offered to people when not in use by

the owner. The owner gets the fees from the customer. Platforms such as zipcar [47], Turo [48], Getaround [49], and Snappcar [50] are also called multi-sided platforms that bring together the different but dependent parties. There are around 65 carsharing companies worldwide that offer different types of services. The underlying structure of P2P networks, various stakeholders, and multi-sided platforms will be transferred to use distributed ledger technologies (DLT) to improve these businesses with independence, trust, and privacy.

Traditional carsharing companies such as Share Now use GPS and geofencing capabilities to define so-called business areas where cars can be picked up and parked. These areas are usually not specified in P2P carsharing offerings.

Carsharing and the GDPR

In [51], the European Data Protection Supervisor gives an example of a common user expectation in the domain of carsharing.

> *For example, if I use an app for carsharing I expect that my location is used for me to know where the closest car is parked and that my contact details be used to get in touch with me in the context of the service. This does not mean that, by default, my location and contact details should be sent over to local bike sellers to send me advertising and offers.*

The example highlights that a user can expect data protection by default, as proposed in the GDPR, which means that the service provider uses only the data necessary for the use case process. This view can be extended to limit further the use of a data subject's location data while using a shared car. The service does not need to know the car's location during the rental.

Limitation of business areas

In a carsharing scenario, the car owner, either a company or a private person, may want to limit the area in which the customer can use it. The idea is that car owners define areas where shared cars can be used. Crossing such an area's border can violate a pre-defined rule and increase the rental price per minute. In Berlin, some carsharing companies limit the business area during certain days of the year to prevent the car from being damaged, see Figure 2.1 from former drive-by mobility, now called Miles mobility [46].

Switch drive power mode

In 2019, BMW announced [52] that it wants to use geofencing services to detect when a plugin-hybrid electric vehicle (PHEV) is within an inner-city environmental zone and automatically switch the car to a non-CO2 exposure mode while driving within this boundary. BMW's motivation is to decrease CO2 emissions within inner cities where other measures failed, such as reduced speed and non-driving zones. Also, earning rewards towards free charging and carsharing option is discussed to motivate drivers. This use case fits perfectly with the presented privacy-preserving geofencing approaches.

Rate changes for traffic congestions

Another situation where pre-defined areas can improve privacy is that dynamic conges-

[1]Image source: Information about the changes was sent on April 27th, 2018, to the consumers of the carsharing service by e-mail.

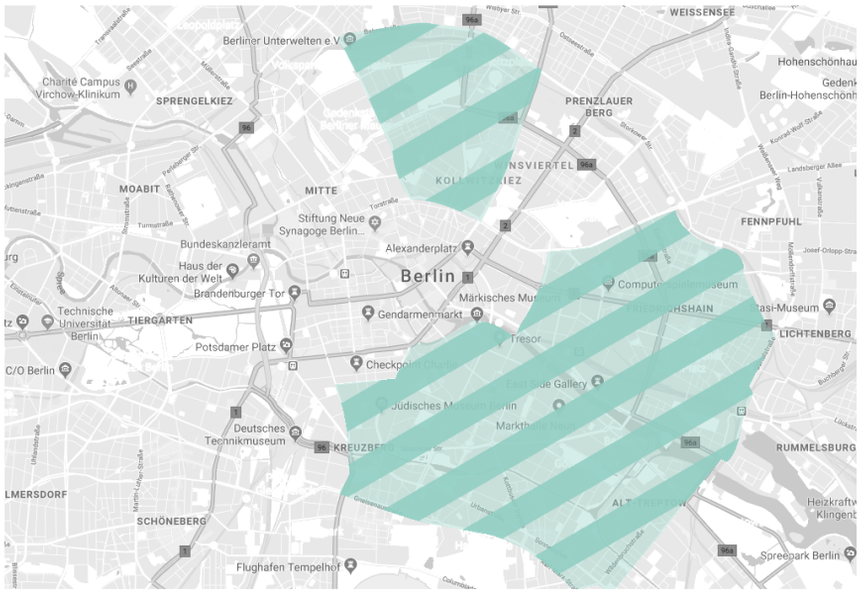


Figure 2.1: Drive-by 72h business area limitations during the May Day demonstrations.[1]

tion areas are evaluated against the car's position. As long as the car is within a traffic congestion, the minute price decreases as an incentive. The objective is that the dynamic areas need to be evaluated against the target client's geolocation in a privacy-preserving way. The challenge is that a low amount of traffic congestion may reveal the user's location, or a service may evaluate it against made-up congestion areas. This approach can also be merged with the distance and/or speed evaluation described in Sections 2.4.5 and 2.4.6. Business models can use this as a selling point, as sharing options are based on time or distance.

2.4.3 Evaluation of cycle paths

The presented solution aims to provide the possibility to check against very accurate geofences with up to centimeter precision. Another possible use case utilizing this precision is evaluating users using specific paths for bicycles, electric scooters, and similar vehicles. A study could be conducted in which these vehicles are evaluated against the paths or roads they take, i.e., streets, footpaths, or cycle paths. As with the previous use case, different geofences could all be represented but would have to be much more accurate in tracking the vehicles' positions. Fortunately, the accuracy of positioning devices is improved by technological advances and other GNSS systems, such as 5G and Galileo.

2.4.4 Remote tolling

Antonini et al. [53] coined the umbrella term *remote tolling* describing the following use cases, also described in [19].

1. Pay-per-use services, e.g., automotive assistance
2. Location-based information services, e.g., traffic information and routes
3. Tracing, tracking, and emergency management services for the transportation of hazardous material and heavy goods
4. Access control services, e.g., enforcement of laws against users who enter restricted

zones such as historic areas without permission
5. Parking services
6. Pay-per-use insurance, where users pay for the insurance of their vehicles based on the percentage of time the vehicle is used, the type of environment in which the driver uses the vehicle typically, and on-board information that can affect the final insurance cost

The recent discussion in Germany may lead to road toll charges, *road tolling* for short, for all vehicles on the street network. Road tolling is a subcategory of the mentioned remote tolling use cases. The solutions described provide privacy for car owners by abstracting from the position on the road by only focusing on the type of road. It can be imagined that the street network is divided into three types of roads, city roads, country roads, and expressways. For each type, there are different toll amounts.

Such a service would need to evaluate how long a specific vehicle moved on each of the three road types, and the driver can be charged accordingly. It is not necessary to know the exact position of the car. The street network could be virtually covered with three types of geofences for each class. One can also imagine charging vehicles only on expressways and evaluating the position against a geofence for only this type of road.

2.4.5 Distance measuring and contact tracing

In the context of the activity or social services, users can upload their training data, e.g., for running or cycling activities. The goal is to compare and share your data with other athletes. There are options to have rankings for specific areas and activities; sportswomen can get suggestions and the ability to contact other athletes nearby. The presented solution aims at providing these functionalities in a privacy-preserving manner. The comparison of running and cycling sportswomen also has to be possible in a privacy-preserving way without disclosing precise location information to such a service. This can mean notifying a sportswoman if another is nearby or suggesting one based on postal code areas. Other social services, such as dating applications, could also benefit. El-Sobhy et al. published a proximity-based service approach to be used in the medical contexts [54]. The following subsection will describe the importance of contact tracing during a pandemic in 2020 as a complementary use case. Anonymous user tracking in *location-based community services* (LBCS) was also researched by Ruppel et al. in 2006 [55] based, for example, on the user tracking and proximity detection work by Küpper and Treu [56] [57] [58] among others.

Contact tracing

The Coronavirus pandemic in 2020 quickly instantiated ideas and applications that should help analyze people's movements who tested positive and inform other people who had been close over a pre-defined time. In China, from where the virus originated, the government informed people via an app about their risk with a green, yellow, and red categorization and advised them to stay in quarantine. In South Korea, they released a similar application to the public [59]. Evaluating this method's usefulness and psychological and social implications is difficult.

Early in the discussion, privacy concerns were raised by researchers and privacy officers at large. They also pointed out the legal and sociological differences between Asian countries and Europe. The approaches described in this thesis will support a privacy-

preserving way of analyzing data for this specific use case.

Distance measuring can actually be applied as the virus is only spread between people. The primary transmission mode is via respiratory droplets that people emit when coughing or sneezing. During the pandemic, experts' advised enacting *social distancing* of 1.5 to 2.5 meters [60]. As the virus is also transmitted via droplets that sit on door handles and other objects, it is also of value to know if people used the same train carriage. Active near-field communication between devices, such as Bluetooth Low Energy (BLE), may also be considered a mode of alarm. It has to be emphasized that all of this must be regarded with the psychological and sociological implications in mind, i.e., *false positive* or *false negative* information can significantly impact people's behavior and feelings.

2.4.6 Telemetry data

As Section 2.4.4 mentions, pay-per-use insurance could monitor a car's telemetry data and charge the driver accordingly. It will be seen that the geofencing approach of this thesis can be used to detect the speed of a target client in general or in specific areas, such as expressways or a traffic-calmed area. An insurance company may evaluate driving style with a privacy-preserving speed evaluation.

2.4.7 Distributed ledger technologies

The advent of blockchain technology is leading to new opportunities for pervasive computing. With decentralized smart contracts that can self-enforce, novel application scenarios and business models emerge [61] [62]. These typically cut out the middleman that traditionally mediated between parties. To be of value, smart contracts frequently need trusted information about the outside world. For example, a smart contract designed to insure against flight delays needs information about delays as they happen in the real world. However, a smart contract cannot simply request this information from a web server on its own. All miners would need to perform the request on behalf of the contract, retrieving the same result. This would also have to be reproducible by anyone at any time. As this is impossible, such outside information can be submitted into the blockchain through prediction markets, where participants vote on a question posed by an inquirer or through so-called oracles, that are single entities claiming that certain events occurred in the outside world. Once inserted, smart contracts can then make decisions based on the available information.

Facilitating a use case such as a decentralized ride-sharing service would be beneficial if a user's location information was accessible to a smart contract. Assuming such a smart contract is designed to charge a user based on time and distance automatically traveled, information about the start and end location would be needed. Regarding sharing a car, it may be interesting to restrict the allowed driving area, as addressed before, geographically. For example, a smart contract could enforce financial penalties if a specified geographic area is exceeded or the car has not been returned to a specific location in time.

While these are specific examples in which reliable location data may be helpful, many other scenarios would pose an advantage, such as verifying the origin or processing locations of products and materials in a supply chain or logistics for cargo tracking delay verification. The approach can also be used in smart cities to authorize social benefits or

Table 2.1: Apps, OS, products and their location privacy settings and features

Apps, OS	Setting or feature	Method category
Strava	privacy zones, limit receivers	spatial, clique
WhatsApp	limit sharing time, limit receivers	temporal, peers
Signal	send current location, limit receivers	temporal, peers
Threema	send current location, limit receivers	temporal, peers
Glympse	data is disassociated from ID after 48 hours	anonymity
ADAMS	possibility to define granularity of information	spatial cloaking
CoronaWarnApp	client-side evaluation	protocol
iOS	grant location access never, once, in use	temporal
FindMy	limit receivers	peers
Airtags	sound signal	protocol

restrict local currency distribution.

However, obtaining reliable location data for a given user or device is challenging. If a user were to submit his position into the blockchain by himself, there wouldn't be anything preventing him from being dishonest. Prediction market participants likely do not know the event's details the requested location pertains to, nor would they be able to act timely.

It will be shown how geofences can be implemented in smart contracts while keeping user information private.

2.5 Location privacy in current apps

Table 2.1 lists apps and operating system and their respective privacy settings and features regarding user locations, together with their method categories. Privacy method categories are also listed in the related work chapter in Section 4.3.

2.6 Prerequisites, objectives, and requirements

This section summarizes the prerequisites, the main objectives, and non-functional requirements as a basis for the upcoming chapters. The author is aware that some of these assumptions need to be addressed by additional software and hardware solutions, such as relying on very precise positioning techniques of a mobile device.

There are two parties involved, the mobile device, i.e., the target client belonging to a data subject, and a server, administered by a service (provider), i.e., the data controller/processor.

2.6.1 Prerequisites

The following five prerequisites are assumptions related to the described use cases and scenarios valid for all objectives.

Prerequisite 1 The mobile device, i.e., the target client, calculates the most accurate and precise position and the service is trusted that the encoded position is accurate.

Prerequisite 2 The data exchanged between the mobile device and the server is secured from any third party eavesdropping, i.e., by data or channel encryption.

Prerequisite 3 There may be an initialization/setup phase between the target user/client and the LBS service. In its most universal way, there is one target user/client and one service evaluating the position against any number of geofences.

Prerequisite 4 The service may store a client identifier together with the geofence information to relate a user to a specific client.

Prerequisite 5 The involved parties follow the protocol but may gain indirect information by analyzing all available data. This security assumption is also described as *honest but curious* and is discussed with other approaches in Section 3.3.

2.6.2 Privacy objectives

The proposed solutions will address the following privacy objectives.

Objective 1 The client sends data only to the service(s) the target user has decided upon. No third party should be involved, such as (additional) location providers or trusted third parties used to do computations on encrypted data or send evaluation results to subscriber services.

Objective 2 The target's position should be as precise as possible, up to a cm level, which may be necessary for specific use cases, such as cycle path evaluation, autonomous driving based on HD maps, and contact tracing. This objective also depends on the positioning technique, which is not addressed in this thesis.

Objective 3 The calculated position and data sent to the service must be computed rapidly for use cases depending on rapid location updates, i.e., fast-moving clients.

Objective 4 The service learns whether or not the target client is within a specific area. No additional information, including information about speed or relative movement, should be revealed.

Objective 5 In certain use cases, it is necessary to hide the client-defined geofence from the service, e.g., in the doping control scenario.

Objective 6 The service may check the quantity and sizes of the areas created by the client. The service should not be able to retrieve information about their locations and the reasons behind them, i.e., a relation to a medical facility. This applies to use cases such as the doping control scenario where athletes create personal privacy areas.

Objective 7 The solutions should also be usable for private-preserving distance measuring between different target users without revealing any information about their positions to a service. This holds for the sports social network and contact tracing services.

Objective 8 Some use cases require showing the target users' precise position if they are outside the defined (privacy-preserving) geofence(s), for example, in the doping control use case. The position should be hidden if the target client is inside a geofence and revealed if the target is outside. In the case of the doping control use case, a DCO is triggering the target client's (eves) re-active location request.

Objective 9 The geofence(s) can have any arbitrary shape up to a centimeter-level precision. The use case and the position technique will need to match the precision level of the geofence. An example is the cycle path evaluation. The solution should be use case agnostic, e.g., the geofences representing streets, cycle paths, postcode areas, city and district boundaries, and other arbitrary shapes. It is not limited to any geometrical form, such as a square or circle.

Objective 10 The geofence storage memory sizes should be feasible, i.e., realistic, for current hardware systems, as of 2022. Geofences are considered to be stored on the service side to support low-tech clients, such as IoT devices, for example, the eves devices within the athletes' scenario. It may be necessary to store geofence information for each user separately.

Objective 11 The geofences held by the LBS service should not be subject to regular updates. The user-geofence relation should be static. Attacks and correlations by linking information over time make this objective challenging to achieve.

Objective 12 The setup phase in which the target client and service agree on the geofences used in the LBS can be performed with the user agent/target client of her choice, e.g., a web browser. The user is not bound to the capabilities of the client.

Objective 13 It may be desirable to update (add or delete) parts or complete sets of geofences. An update should not reveal any additional information. Sizes and number of areas may be checked before an update, see Objective 7.

Objective 14 The service can decide on-the-fly what kind of geofence will be evaluated against the client's position. This can be useful if there is a scenario in which the geofence area is variable, e.g., if the service would charge less for a car sharing rental in case of traffic congestions.

Objective 15 For an evaluation, such as the described cycle path use cases, a low-tech device may record all the positions taken during a day. These positions could be aggregated and sent later to an evaluation service. That means the low-tech device recording the position does not necessarily need to compile the encoding. A second device, e.g., a smartphone, could do the more computationally heavy work.

Objective 16 The service must not know any additional information about other target clients, such as secret key values. The service should not be able to obtain those while acting as a user. This also means that two targets should only know another target client's position if nearby.

Objective 17 Use encodings and privacy approaches that work with indoor use cases and positioning technologies, such as Bluetooth (Low Energy) beacons or Wi-Fi signals.

The 17 objectives are evaluated and summarized in various tables throughout the thesis, starting with Table 2.2 showing the objectives for each of the eight use cases previously described in Section 2.4.

Table 2.2: Privacy use cases and objectives

	1: Data send to service	2: Precise target position	3: Fast encoding	4: Service learns what's necessary	5: Geofence not known to service	6: Service may limit geofences	7: Privacy-preserving distance measuring	8: Position shown if outside	9: Arbitrary shaped geofence	10: Feasible memory sizes	11: Geofence updates not necessary (privacy)	12: Low-powered device	13: Geofences may be added, deleted, or updated	14: On-the-fly geofences	15: Separation of positioning and encoding	16: Services cannot get shared user secrets	17: Support BLE, Wi-Fi, and other identifiers
1 Whereabouts	●	◐	●	●	●	●	○	●	●	●	●	●	●	○	○	●	●
2 Carsharing	●	◐	●	●	○	○	○	○	●	●	●	●	○	●	○	●	●
3 Cycle path evaluation	●	●	○	●	○	○	○	●	●	●	●	○	○	○	●	●	●
4 Toll charge	●	◐	●	●	○	○	○	○	●	●	●	○	○	●	○	●	●
5 Social network distance	●	●	●	◐	●	◐	●	●	○	●	●	●	●	○	○	●	●
6 Contact tracing	◐	●	○	●	●	◐	●	○	○	●	●	●	●	●	◐	●	●
7 Speed measuring	●	●	●	●	○	○	○	○	○	●	●	●	●	●	◐	●	○
8 Geofences in DLTs	○	●	●	●	●	◐	●	○	●	●	●	●	◐	◐	●	●	●

● Objective is applicable to use case
◐ Objective is partially applicable to use case
○ Objective is not applicable to use case

2.6.3 Non-functional requirements

These non-functional requirements mainly aim to introduce computationally feasible solutions with current hardware and devices.

Non-functional requirement 1 The transformation of a geofence into a cryptographically encoded geofence can be computed on the (mobile) device of the target user who creates it, e.g., a mobile device app or a browser on a laptop.

Non-functional requirement 2 The calculation of a position against a large number of areas is acceptable and suitable for state-of-the-art location-based services.

Non-functional requirement 3 The application for creating a geofence is designed to be user-friendly.

Non-functional requirement 4 Creating a large number of areas and their cryptographic encodings on the server-side are done in an acceptable time.

An extensive (computational) evaluation is given in Section 7.3.

2.7 Goals

This section summarizes the main points of the presented thesis, including the *thesis statement*, *research questions*, *contributions*, and *aspects not addressed* by it.

2.7.1 Thesis statement

Following the topic's motivation based on various described use cases, the thesis statement is as follows.

- One can design a geofence-based service with centimeter accuracy that respects the user's location privacy from a service provider without the need to introduce a trusted third party.

2.7.2 Research questions

The thesis concept and design contribution address the following research questions.

- How can a geofence with an arbitrary shape be encoded?
- How can the position be checked against the encoded geofence?
- How can the encoding guarantee evaluation of exact positions?
- How can location privacy be achieved even knowing the user's identity?
- What methods exist to evaluate a masked position against a masked geofence?
- How can the introduction of a trusted third party be avoided?
- What additional features can the privacy-preserving mechanisms and geo-encodings address?
- Can the presented approaches run on current hardware?
- How could services and mobile OS providers use the method?
- What are the differences between LBS classes?

- How are these methods categorized in a general location-privacy taxonomy?
- What are attack vectors, and how can they be avoided?

2.7.3 Contributions

The thesis outcome provides the following contributions to the state-of-the-art research.

- Evaluation and proposition of encoding systems for geofences and positions with arbitrary shapes.
- Novell approach of position and geofence evaluation down to centimeter-level accuracy.
- Introduction of inverted geofences.
- Evaluation and proposition of different masking methods, including the grid cell hashing method.
- Evaluation and proposition of how these methods would be feasible in real-world scenarios.
- Privacy-preserving distance and speed measuring.
- Extension of the location privacy taxonomy.
- Overview of attack vectors and their relation to different privacy-preserving approaches.

Chapters 3 and 4 will overview related work and state-of-the-art technology. Some technology/research questions are only addressable while establishing and extending current underlying technologies, protocols, and standards. It would also require hardware alterations to solve some of these open issues.

Using positioning information in the context of security and privacy quickly raises questions about that information's genuineness. The broader term *proof or presence* and the more restricted term *proof of location* are used when discussing these concepts. The thesis addresses these terms briefly. Recent developments in 5G [63] and GNSS systems, such as Galileo, show that solutions require hardware, software, and protocol changes to make location spoofing detectable or impossible at best.

Accuracy is a fundamental cornerstone of location-based applications besides ensuring that the received position is not tampered with. Client positioning methods will improve over time to enable very high accuracy geofencing levels.

2.7.4 Out of scope

The thesis does not contribute to these particular location and privacy-related research areas.

- Improving positioning accuracy
- Location spoofing resistance
- Advances in user anonymity or pseudonymity
- Position blurring as a privacy method
- Specific measures for hiding user positions from intelligence agencies

It is important to remember the out-of-scope aspects to avoid getting confused by the capabilities of cryptographic solutions.

2.8 Research methodology

The *design science* approach is a problem-solving paradigm described by Hevner et al. in [64] [65]. In contrast to *behavioral science* paradigms, design science aims to create innovations that define ideas, practices, technical capabilities, and products. The goal can be efficiently and effectively accomplished by analysis, design, implementation, management, and the use of information systems [66].

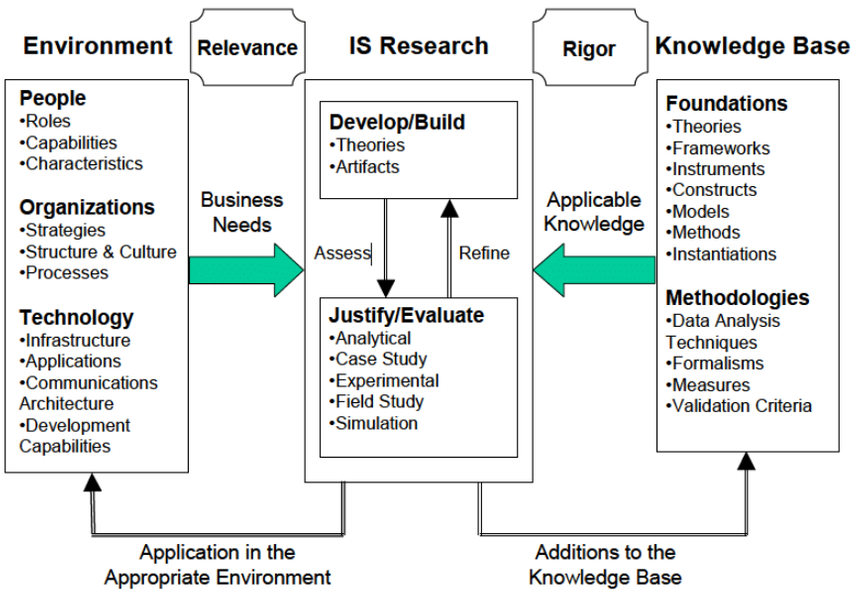


Figure 2.2: Research framework for information systems.[2]

Design science focuses on creating (IT) artifacts, including the following forms.

- **Constructs**: Vocabulary and symbols
- **Models**: Abstraction and representation
- **Methods**: Algorithms and practices
- **Instantiations**: Implemented and prototype systems

The overall goal of design science research is utility [64]. He defined a set of guidelines, see Table 2.3, to accomplish this task. The process is divided into three cycles that provide a framework, see Figure 2.2 for activities [65].

Firstly, the relevance cycle describes the definition of the research problem and an assessment of its relevance. Secondly, in the rigor cycle, an evaluation of related work is conducted to assure the artifact's novelty is to be constructed. Thirdly, the design cycle implements and evaluates the artifact(s) and provides a detailed and formal description of each result.

Chapter 2 motivates and assesses the research being conducted in this thesis (*relevance*). Chapters 3 and 4 analysis the state-of-the-art (*rigor*). Chapter 5 provides a detailed description of the contribution, detailing *design decisions* of different approaches. Chapter 6 gives an overview of the implemented *artifacts*. Chapter 7 evaluates the artifacts and shows the *improvements* to the related work (*design*). The design science evaluation for the constructed artifacts are summarized in Table 7.1.

[2]Image source: Information Systems Research Framework, Hevner et al., 2004.

Table 2.3: Design science research guidelines

Guideline	Description
1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

2.9 Chapter summary

The chapter defined the different notions of privacy and linked them to legal aspects. It introduced different privacy approaches and use cases where privacy-preserving location-based solutions are added value. From the application scenarios, the objectives and requirements were derived. The chapter also presented the research questions and the methodological approach to generate innovative solutions.

There's no place like home.

The Wizard of OZ (1939)

3 Definitions

The scenarios described in the motivation chapter gave an overview of where privacy-preserving location-based services can be applied to increase users' freedom. The following definitions are given on terms, roles, actors, and models. An overview of security definitions, privacy objectives, attacks, and countermeasures is also provided.

3.1 Location-based services

In general, this thesis focuses on various location-based services (LBS). How LBS evolved from the mid-90s to the mid-2010s can be found in [67] [68]. The user of an LBS is called *target user (TU)*. A target user can be a natural person, a small *IoT device*, or a larger thing, such as a car. A *mobile terminal (MT)* can locate itself, e.g., by A-GPS positioning, and is referred to as *terminal-based positioning* [69]. Positioning methods, including the mobile network operator, a third-party provider, or server components, are summarized as *network-based positioning*. Both approaches are also called *device- and network-centric positioning*. A mobile terminal can be, for example, a smartphone, laptop, or dedicated GPS device. They are named *target clients* or just *clients*.

Encoded position data is usually sent to a backend component. Such servers are *location servers* (LS). They include databases or services related to the positioning evaluation and provide location-based services. Whether these components are considered trustworthy or not is defined in Section 3.2.

LBS are subdivided into *proactive* and *reactive* services. The MT frequently performs a *position update (PU)* and actively sends or processes its position in proactive services. In reactive services, the MT's position update is triggered remotely by an LBS or LS, referred to as *polling*. Other forms of PUs are *immediate*, *periodic*, *distance-based*, *zone-based*, and *piggybacking*.

If a user's location is processed for her purpose, the LBS is *self-directional*. If a user tracks another target client's position, the LBS is *uni-directional*, e.g., doping-control

whereabouts. In the case of a location-based social network, such as Strava or contact tracing apps, the term *bi-directional* was coined.

The different technologies for positioning target clients are mainly categorized into *indoor* and *outdoor* variants. Outdoor positioning is done via GNSS services, such as GPS or Galileo, Cell-IDs, and Wi-Fi positioning. Indoor LBS are usually relying on Wi-Fi and Bluetooth positioning techniques. These techniques depend on methods such as angle of arrival (AOA), time of arrival (TOA), time of flight (TOF), and time difference of arrival (TDOA). More recent positioning techniques utilize ultra-wideband radio signals and the Earth's magnetic field.

In this thesis, the described use cases depend on accurate positioning, mainly for outdoor scenarios. As the positioning evaluation is abstracted, the privacy-preserving approaches can also be transitioned to indoor LBS. The precision level can also be adjusted to the use case requirements. Table 3.1 summarizes the main classification concepts of LBS services.

Table 3.1: Overview LBS classification attributes

Classification concepts	Values
Triggering/Invocation	proactive, reactive
Direction	self-directional, uni-directional, bi-directional
Surroundings	indoor, outdoor
Measurement mode	network-centric, terminal-centric
Operation architecture	client/server, peer-to-peer
Measurement references	observable, position calculation, descriptive, spatial, infrastructure
Positioning methods	proximity sensing, lateration, angulation, dead reckoning, pattern matching
Observables	Cell-ID, coordinates, range (difference), angle, direction, velocity, distance, visual images, fingerprints, Wifi signals, RFID
Infrastructure networks	Satellites, Cellular, RFID, Bluetooth beacons, Wifi, terminals
Quality parameters	Accuracy, precision, yield, consistency, signaling and computational overhead, power consumption, latency, roll-out, and operating costs

3.1.1 Geofences and geofencing

The coined terms geofence and geofencing will be adapted for the thesis' context. Figure 3.1 shows a circular geofence service abstraction with entering, dwelling, and exiting events from the Google developer API documentation [70].

The terms' most common meanings are described with these two definitions for nouns and verb [71].

geofence / geofencing

noun | ˈdʒiːoʊfens |

> *A virtual geographic boundary, defined by GPS or RFID technology, that enables software to trigger a response when a mobile device enters or leaves a particular area.*

verb [with object]

> *Create a virtual geographic boundary around (an area) by means of GPS or RFID technology, enabling software to trigger a response when a mobile device enters or leaves the area.*

Within this thesis, the term *geofence* is understood as an arbitrarily shaped polygon that describes a virtual boundary of GPS points in the physical world. These areas and their virtual boundary can represent city or postcode edges, hospital grounds or cemeteries, small areas such as doctor's or psychotherapist's office buildings, or large areas such as street networks of an entire country.

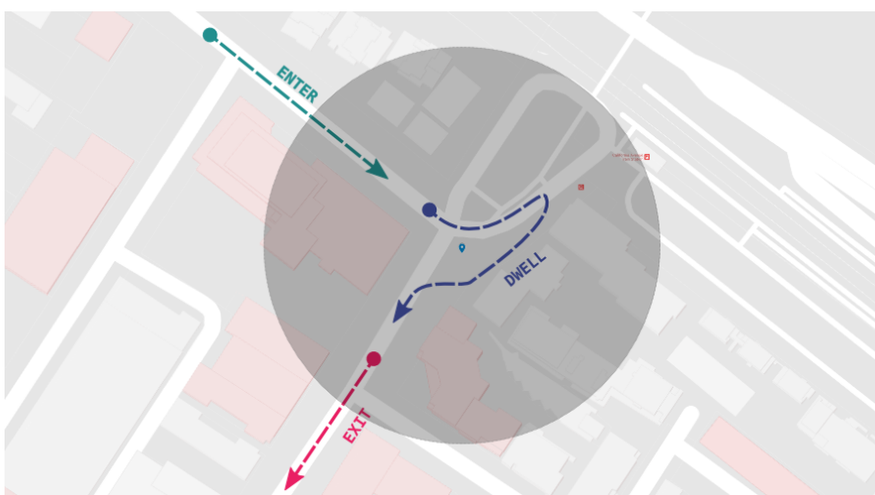


Figure 3.1: Geofencing abstraction from Google API.[1]

The term *geofencing* indicates that an LBS application depends on input and output values on the positioning of a target client and the evaluation against one or more *geofence areas* [72]. The term geofencing in the literature refers mainly to proactive services [73] [74] [75] [76] [77]. In this thesis, the terms *geofence* and *geofencing* are used for any LBS, *proactive* and *reactive*. Garzon et al. [78] coined the term Geofencing 2.0 for a broad range of notification systems, not only based on location aspects. They also categorized geofences into spatial, hierarchy-based, network-based, and semantic types.

3.1.2 From use cases to classes

The thesis focuses on the following LBS attributes grouped into three classes (1-3). The attributes valid for all classes are given in the last row in Table 3.2. These three classes abstract from the use cases presented in Section 2.4.

[1] Image source: https://developers.google.com/location-context/geofencing.

Table 3.2: Classification of addressed geofences services

Classes	Attributes	Use cases
Class 1	Client generates geofence(s), service may verify geofences, e.g., amount, size, client knows geofence(s), reactive, periodic, uni- and bi-directional, infrastructure-assisted	Whereabouts
Class 2	Service generates geofence(s), client knows geofence(s), service knows geofence(s), pro- and reactive, periodic, uni-directional, mobile-based, infrastructure-based, infrastructure-assisted	Carsharing, cycle paths, toll charge
Class 3	Client generates geofence(s), service may verify geofences, e.g., amount, size, client knows geofence(s), reactive, bi-directional, mobile-based	Social network distance, contact tracing, speed measuring
For all	Coordinates, terminal-centric, high accuracy, high precision, yield, high consistency, low latency, client/server-based operation, peer-to-peer-based operation (for specific use cases, such as location-based social networks and contact tracing)	all

3.2 Roles and actors

As defined in [79], an *actor* is *an autonomous entity* (e.g., a person, a company, or an organization) that adopts one or several *roles* that characterize either the *functions* it fulfills from a technical point of view or the *impacts* it exerts on LBSs from an economic or regulatory point of view.

Based on Frattasi et al. [19], two main categories of actors are specified. The *operational actors* are directly involved in LBS's practical operation and functioning. The *nonoperational actors* have an indirect influence on the overall function of LBS.

3.2.1 Roles

As mentioned, the roles describe a (technical) function or their applied impact within an LBS. The following roles and actors are partially adapted from [79] and [80].

- **User**: A person who requests or consumes an LBS also sees the target user below.
- **Natural person**: In jurisprudence, a natural person is a person, in legal meaning, i.e., one who has its legal personality, an individual human being. The GDPR adopted the same term, see Section 2.2.
- **Natural person satellites**: Persons sharing the same location, such as family members, friends, and colleagues. But also devices, such as IoT, cars, bikes, or smart speakers [81].
- **Target or target user (TU)**: An individual to be located.
- **Target client or terminal**: The target's device, e.g., a smartphone or mobile device.
- **Location provider (LP)**: A (server) component which evaluates location policies, also called *location server*.
- **Service provider (SP)**: A provider operating server components, supplying services based on location information.
- **Trusted entity (TE)**: An actor allowed to know the target's actual position, e.g., the

smartphone [82].
- **Untrusted entity (UE)**: An actor not allowed to know the target's actual position, e.g., the LS of an LBS [82].
- **A partially trusted entity (PTE)**: In the semi-honest security model, the participating entities follow the protocol but try to gain additional information by correlating data.
- **Network operator (NO)**: A service provider maintaining a mobile network and services for mobile terminals, including call forwarding and internet protocol (IP) services.
- **Hardware manufacturer (HM)**: Company that manufactures target clients, i.e., mobile devices.
- **Operating system (OS)**: A software enabling the usage of LBS services, i.e., mobile applications.
- **Operating system manufacturer (OSM)**: A company that develops an OS [80].
- **Mobile application (MA)**: A software that runs on top of the OS, i.e., the core application [80].
- **Mobile application developer (MAD)**: A company or single developer that develops an MA.
- **Third party software (TPS)**: Any additional software that is integrated with the MA [80].
- **Third party software developer (TPSD)**: A company or single developer implementing the TPS.
- **Location-based service (LBS)**: A service utilizing a target client's geolocation using the hardware capabilities, the network, and the OS and TPS software interfaces.
- **Location-based service provider (LBSP)**: Legal and technical entity running the LBS [80].
- **Government (Gov)**: Any governmental institution and unit, e.g., intelligence agencies or law enforcement authorities, with legal rights to access companies' databases or communication infrastructures.
- **Prover (P)**: An entity that proves something, e.g., knowledge or possession, to a verifier. In a zero-knowledge proof, the transferred data does not reveal the knowledge and is therefore used in privacy and security use cases, sometimes called a *holder* (of credentials).
- **Verifier (V)**: An entity that receives a proof and validates it.
- **Trusted third party (TTP)**: An entity that gets to know some of the geofence evaluation information, see TE.

From this enumeration, the following roles are presented in Figure 3.2, *user, natural person, target user, target client, (location-based) service provider*, and *trusted third party*.

3.2.2 Actors

The following actors are used to illustrate the use cases and different privacy approaches. Some of these are directly put into relation in Figures 3.2 and 3.3.

- **Target users, e.g., Andy and Carrie**: Targets to be located, users requesting or consuming an LBS, in case of a self-directional LBS.
- **Service provider operator, e.g., Becky**: A semi-honest entity. Together with the provided service, Becky and EDEN are verifiers of the target user's location.

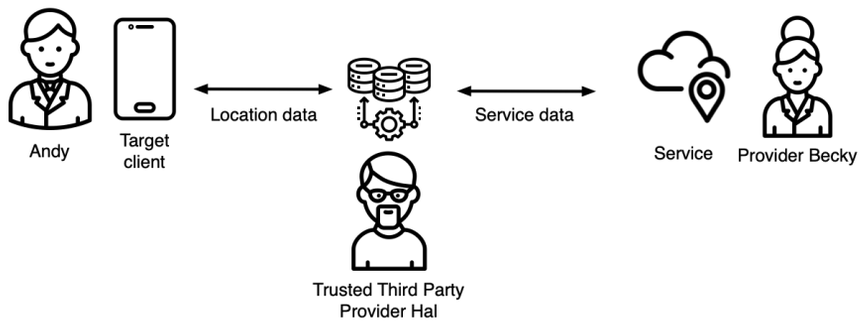


Figure 3.2: LBS roles and actors.

- **Service, e.g., EDEN**: An LBS service operated by Becky and her employees. Target users use this service.
- **Target client smartphones**: These belong to (target) users, i.e., Andy and Carrie.
- **Target client, e.g., eves device**: A dedicated positioning device with network capabilities.
- **Trusted third party service**: In some approaches, a trusted third party is required for a privacy-preserving evaluation. This service may know the location of a target but has no additional information.
- **Trusted third party service provider, e.g., Hal**: The person operating the TTP. This person must prevent information leakage.
- **Adversaries (untrusted entities), e.g., Eve, Ellie, and Carl**: In a classical security model, individuals or organizations try to get access to private data and information of services and users.
- **Key Authority**: A key authority is trusted with generating public and secret keys, as well as holding and distributing them.
- **Mobile operator**: The operator of the mobile network that the target client uses.
- **Operating system provider**: A provider, such as Google or Apple, provides the APIs for their respective operating systems.
- **GNSS operator**: An operator of a global navigation satellite system, such as GPS, Galileo, GLONASS, or Beidou.

From this enumeration, the following actors are presented in Figure 3.3, target user *Andy*, target client smartphone, and (location-based) service provider *Becky*. In this figure, the actor Becky is also the LBS *adversary*.

3.2.3 Adversaries

An adversary tries to access the target user's location information that can be sold or used. The privacy adversaries either aim at the client or the LBS, see Figure 3.3. In an access control scenario, an adversary wants to pretend to be at a location she is not to gain access to data or a service. Section 3.4 lists types of attacks and their countermeasures.

The following list provides an overview of possible attackers within an LBS.

- **LBS Provider**: The provider itself is the most likely adversary in these scenarios. It gets to know the target's exact location to provide the service.
- **Location Provider**: If the LP is different from the LBS provider, it can maliciously use its target's location.
- **Cloud Provider**: A service just hosting the LBS providers or the LP data and services

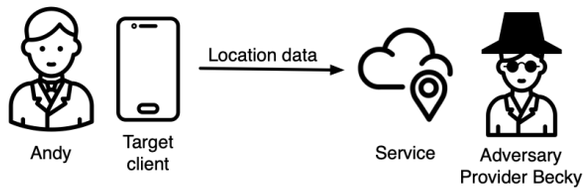


Figure 3.3: Privacy adversary of an LBS service.

can be considered a third-party attacker.

- **LBS users**: In some solutions, the privacy methods depend on a known secret for a community. Within small groups, sharing location data might be useful. This approach might be vulnerable in large communities as known secrets can be used to gain whereabouts information.
- **Malicious Employees**: Provider employees may access the data and the target user's information, maybe celebrities. The attackers can gain private information and sell it.
- **External hackers**: Any person or group who might misuse security flaws within the database and service software might be considering attacking an LBS service provider to steal and maliciously use private data.

It has to be emphasized again that the approaches are not particularly targeted at hiding personal information from national and international intelligence and security agencies. These entities have access to technologies targeting operating system exploits. The goal is that an agency with access to the provider's data does not gain more information than the provider itself.

3.2.4 Identifiers

Based on the location privacy work of Pfitzmann et al. from 2010 [83] and Georgiadou et al. in 2019 [81], the following central data sources are identified.

- **Unique identifier**: This is a data element associated with just one entity of interest. That entity may be a data subject or something else [81].
- **Key identifier**: A data element that can be exploited with minimal effort to identify a (privacy-sensitive) data subject [81].
- **Quasi identifier**: A data element that almost discloses a data subject's identity due to its semi-unique value and will allow full disclosure when combined with other quasi-identifiers [81].
- **Private attribute**: The remainder class of privacy-relevant but non-identifying data elements [81].
- **Item of interest**: Within the context, the general term item(s) of interest (IOI) is used to describe valuable information for an adversary, such as subject-IDs, messages, actions, data, knowledge, and information bits. Pfitzmann and Hansen define *linkability* of IOIs as follows. *Linkability of two or more items of interest (IOIs, e.g., subjects, messages, actions) from an attacker's perspective means that within the system (comprising these and possibly other items), the attacker can sufficiently distinguish whether these IOIs are related or not.* [83]. The countermeasure is *unlinkability*, see Section 3.3.6.4 for more details.

Figure 3.4 represents the privacy spectrum scope with relation to *personally identifi-*

able information (PII), taken from the W3C standardization of decentralized identifiers [84]. Unique and key identifiers belong to the spectrum's *highly correlatable* part, quasi-identifiers belong to the *correlatable part*, and private identifiers to the part of *non-correlatable* information. For an adversary, all items are considered of interest.

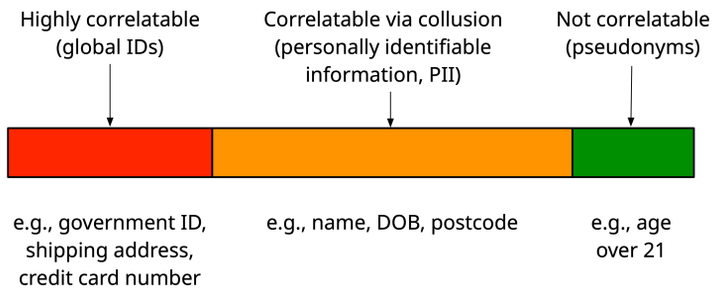


Figure 3.4: Privacy spectrum ranging from pseudonymous to fully identified.

3.3 Security and privacy models

The following subsections present the relevant security definitions. The privacy-preserving approaches aim at making a trusted third party obsolete. There are only two parties involved, the target client and the (location-based) service.

3.3.1 The real-world and ideal-world models

In the *ideal-world model*, an incorruptible trusted (third) party exists to which each protocol participant sends its input values. This trusted party computes the protocol output and sends the result back to the entities, see Figure 3.2 [85].

In the *real-world model*, there is no trusted (third) party, and all entities involved in the security protocol can exchange messages.

A protocol is considered secure if each entity cannot learn more about other entities' private inputs than each of those could learn in the ideal-world model. In the ideal-world model, no messages are exchanged among the entities. In the real-world model, any message sent must not reveal information to another party.

Depending on the security or privacy protocol, adversaries can be categorized into groups depending on their willingness to deviate from a protocol. *Semi-honest (passive)* and *malicious (active)* are the primary security model classes. The semi-honest model builds the foundation of this thesis [86].

3.3.2 Semi-honest (passive) security

The entities follow the protocol in the semi-honest or passive security protocol, i.e., they do not deviate from it. Nevertheless, they collect all data (IOI) and infer information. This is the *naive* advisory model and is regarded as the weaker security solution. But these protocols have two advantages. First, they can be applied to privacy use cases where the entities should learn nothing about the specific data from another participant. Secondly, they are efficient in real-world scenarios and form the basis for higher levels of security, i.e., the *active security model* [87] [86].

Within the privacy use cases, we consider all involved parties semi-honest, which means that the parties follow the privacy-preserving protocols and should not gain any additional information about private data if not specifically intended.

Cryptographically speaking, it is assumed that adding a random number r to x effectively hides x, and in real number scenarios knowing the sum of two real numbers $r_1, r_2 \in \mathbb{R}$ reveals nothing about both r_1 and r_2 [87] [88].

3.3.3 Malicious (active) security

In the malicious or active security model, the involved parties deviate from the protocol execution to learn other participants' secret information. Protocols addressing this behavior are considered to be of higher security. However, this goal is much more challenging to achieve [87] [86].

3.3.4 Semantic security

In a *semantic secure* cryptosystem, an adversary can only extract negligible information about a message (plaintext) from a given ciphertext. This notion was included in the definition made by Goldwasser and Micali in 1982 [89]. Today the definition is equivalent to the *ciphertext indistinguishability* under the chosen-plaintext attack. Given two plaintexts and the corresponding ciphertexts, an attacker cannot distinguish which plaintext belongs to which ciphertext [90].

3.3.5 Cryptography objectives

The following cryptographic objectives are usually associated with security requirements of (web-based) communication protocols between two users.

3.3.5.1 Confidentiality

User data should be confidential within both security and privacy contexts, i.e., to an adversary who wants to know detailed secrets about a user. Data encryption with a robust algorithm within security settings, which should only be visible to the parties who hold a key for decryption, ensures confidentiality. Within a privacy setting, keeping the user data confidential from any unwanted party and consuming a service is also done by applying cryptography.

3.3.5.2 Authentication

In the described replay attack in the context of user authentication, there are two common countermeasures [90]. The first solution includes the name and a time-stamp of the sender of the authenticated message. The second solution consists of a random number chosen by the verifier to be included in the message signature. The random number is different for any new message/authentication. Such a number is called a *challenge* and the whole approach is a *challenge-response* identification.

3.3.5.3 Integrity

Integrity is a property possessed by data items that have not been altered in an unauthorized manner since they were created, transmitted, or stored [91].

3.3.5.4 Non-repudiation

Assurance that the sender is provided with proof of delivery and the recipient is provided with a proof of the sender's identity so that neither can later deny having processed the data [92].

3.3.5.5 Availability

Ensuring timely and reliable access to information is the availability objective. Business resiliency objectives extend the concept of availability to refer to a point-in-time availability, i.e., the system, component, or device is usable when needed, and the continuity of availability, i.e., the system, component, or device remains functional for the duration of the time it is required [93].

3.3.6 Privacy objectives

Additionally to the security objectives, the following requirements are associated with privacy-preserving aspects of communication protocols, mainly derived from legal considerations as described in Section 2.2.

3.3.6.1 Minimization

Data processing should only use as much information as required to successfully carry out a given user's service. This means that services must limit personal data collection, storage, and usage to relevant, adequate, and absolutely necessary data for carrying out the purpose for which the data is processed. This is one of the key measures stated in the GDPR [24].

3.3.6.2 Intervenability

Intervenablitilty is defined as the property that intervention is possible concerning all ongoing or planned privacy-relevant data processing [94].

3.3.6.3 Transparency

Transparency is the property that all privacy-relevant data processing, including the legal, technical, and organizational setting, can be understood and reconstructed at any time [94].

3.3.6.4 Unlinkability

The terminology of unlinkability described by Pfitzmann and Hansen is being used within this thesis [83]. They define *unlinkability* and the *unlinkability delta* as:

> *Since we assume that the attacker does not forget anything, unlinkability cannot increase. Therefore, the unlinkability delta can never be positive. Having an*

> *unlinkability delta of zero means that the probability of those items being related from the attacker's perspective stays exactly the same before (a-priori knowledge) and after the attacker's observations (a-posteriori knowledge of the attacker). To be able to express this conveniently, we use wordings like "perfect preservation of unlinkability w.r.t. specific items" to express that the unlinkability delta is zero.*

3.3.6.5 Self-Sovereignty

The concept of self-sovereignty is currently discussed in the context of identity information. A holder can store and control her own verifiable credentials [84] without relying on a trusted third party. This is known as self-sovereign identity (SSI) and utilizes approaches such as distributed ledger technologies (DLT), zero-knowledge proofs, and blind signatures [95]. In the presented work, a user is enabled to limit the service's access to private location information by empowering her with privacy-preserving mechanisms.

3.4 Attacks and countermeasures

In 1883, Auguste Kerckhoffs stated six design principles for military ciphers, the first formulation of a fundamental assumption in cryptoanalysis [96]. The English translation reads as follows [97].

1. The system must be substantially, if not mathematically, undecipherable;
2. The system must not require secrecy and can be stolen by the enemy without causing trouble;
3. It must be easy to communicate and retain the key without the aid of written notes, it must also be easy to change or modify the key at the discretion of the correspondents;
4. The system ought to be compatible with telegraph communication;
5. The system must be portable, and its use must not require more than one person;
6. Finally, given the circumstances in which such system is applied, it must be easy to use and must neither stress the mind or require the knowledge of a long series of rules.

In modern cryptography, Kerckhoffs' principle is summarized as follows.

A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

In 1949 the American mathematician Claude Shannon described the same principle as "the enemy knows the system" in the communication theory of secrecy systems [98]. The location-based system provider is the *adversary* or related to Kerckhoff's or Shannon's wording *the enemy*. Often the synonym *Eve* is used to describe a generic adversary.

Stalling [24] describes an attack as follows.

> *Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself.*

In that sense, it is different from the meaning of *thread*, which is described as.

> *Any circumstance or event that has the potential to adversely impact organi-*

> *zational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the nation through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.*

The X.800 recommendations [99] for secure architecture elements for open systems classify attacks as either *passive attacks* or *active attacks*. In a passive attack, the information resource is not affected, but the adversary attempts to gather information instead from the (information) system itself. An active attack alters system resources or tries to affect their operations.

3.4.1 Cryptological attacks

In cryptology, the sum of cryptography and cryptoananalysis, attacks are classified into *ciphertext-only attacks*, *known-plaintext attacks*, *chosen-and adaptively-chosen-plaintext attacks*, and *chosen- and adaptively-chosen-ciphertext attacks* [90].

Ciphertext-only attack In this scenario, the adversary can only access the ciphertext, i.e., an encrypted message obtained, for example, by eavesdropping. An encryption method not able to withstand such an attack is insecure.

Known-plaintext attack If an attacker can obtain plaintext-ciphertext pairs, it can give her an advantage in attempting to decrypt ciphertext for which she *does not* have the corresponding plaintext. This attack is significant for the given context, in which the LBS may get plain and encoded location data from the client.

Chosen- and adaptively-chosen-plaintext attack The attacker can obtain every ciphertext for any chosen-plaintext in this category. With the analyses of these pairs, the attacker might decrypt messages for which she does not have a plaintext. At first glance, this scenario seems very unlikely. However, in the context of public key cryptography, where the encryption key is public, an adversary can get any number of ciphertexts for previously chosen plaintexts.

Eve can only get pairs of texts before her analyses without further interactions. However, as in the public key example, Eve can subsequently get more pairs in an adaptively-chosen-plaintext attack. In both versions, the attacker tries to get hold of the encryption key.

Consider the following extension. After the last step,

1. The attacker has two plaintexts, m_0 and m_1.
2. A bit b is chosen uniformly at random $b \leftarrow \{0, 1\}$.
3. The attacker receives the encryption of m_b, attempts to *guess* what plaintext she received, and outputs a bit b'.

A cipher has *indistinguishable encryptions under a chosen-plaintext attack* if, after running the above protocol with $n = 1$, Eve can't guess correctly $(b = b')$ with probability *non-negligibly* better than 1/2. This indistinguishability is also the aim of modern cryptographic protocols and is referred to as *semantic security* [89].

Security against ciphertext-only attacks in a public key encryption scheme includes

security against adaptively-chosen-plaintext attacks [90].

Chosen- and adaptively-chosen-ciphertext attack This category of attacks is similar to the previous one. The difference is that the attacker can access the decryption mechanism and therefore generate plaintexts from chosen ciphertexts. The acronym CCA1 denotes non-adaptive-chosen-ciphertext attacks (also called *lunchtime* or *midnight attacks*) and adaptively-chosen ones denoted by CCA2. In RSA Security compliant messages, there is also the term *partial-chosen-ciphertext attack*, based on Daniel Bleichenbacher's approach [100] from 1998, notably called the *1-Million-Chosen-Ciphertext Attack of Bleichenbacher*. The attack is based on the fact that PKCS#1(v1.5)-based ciphertexts have a distinct and known padding format used to get secret information about SSL/TLS encryption keys.

In homomorphic public-key encryption algorithms, see Section 4.4.3, there is a property called *malleability*. An encryption algorithm is *malleable* if it is possible to transform an (unknown) ciphertext to another ciphertext, which decrypts to a related plaintext. Given the encryption of a plaintext m, it is possible to generate another ciphertext that decrypts to $f(m)$, for a known function $f()$ without necessarily knowing or learning m [101].

A cryptosystem can be semantically secure against chosen-plaintext attacks or even CCA1 while being malleable. However, a cryptosystem that is not secure against CCA2 is equivalent to non-malleability [102].

Semantic security and indistinguishability are also equivalent in the case of adaptively-chosen-ciphertext attacks, as shown in [103].

Small-message-space attack An attacker can encrypt all messages with a given public key if possible messages are small, and all messages are known in advance. Later, she can decrypt an intercepted encrypted message by comparing it to the precomputed encrypted messages [90]. The message space of encoded location information can be considered small and known in advance.

Common-modulus attack When using the RSA cryptosystem scheme and two users with the same modulus n have two exponents e_1 and e_2, which are relative prime, one can derive the other's secret key due to the scheme's construction. Therefore, prime factors need to be chosen randomly [90].

Low-encryption-exponent attack Suppose several users have the same small encryption exponent e in an RSA scheme. In that case, e.g., $e = 3$ and the same message is sent to all of these users, the modulus m can be calculated easily using the Chinese Remainder's inverse isomorphism [90].

Replay attack An adversary could intercept a signed message in an authentication context and later impersonate the original sender/user. Such an attack is a *replay attack*.

3.4.2 Hashing related privacy attacks

The following attacks need to be prevented with the proposed *grid cell hashing* solutions in Chapter 5.3.1.

Dictionary attack A dictionary attack is used to find hash values for a given input value. An attacker could exploit the hash value for a given password or a given passphrase when trying to break into an online account for a specific user. The name dictionary implies that this method uses common words to compile a list of hash values from a dictionary.

Brute-force attack A brute-force attack is very similar to a dictionary attack. However, where a dictionary attack may use pre-compiled lists of passwords, a brute-force attack uses computationally capable hardware to calculate large amounts of hash values or encryption key values by iterating through input values of strings and numbers to find a matching result.

Lookup tables Nowadays, there are databases with password hash combinations. Access to such data makes computing hash values with dictionary or brute-force attacks unnecessary. The computation time can be saved if the needed values are retrieved from a database. *Reverse lookup tables* and *rainbow tables* are two optimized variants. They store pre-compile value chains, altering between hash and reduction functions. A reduction function usually reduces the hash length, for example, using just the first eight characters (for eight-character passwords). To find a plaintext to a hash, the functions are applied and compared to stored values in the chains until a plaintext password matches.

Length extension attacks In this attack, an adversary can get a hash value of a *padded* message without knowing a particular secret. It can be prevented by applying the hash function twice, see Section 5.3.1.3 for details [90].

Birthday attack The brute-force birthday attack[2] is aimed against collision resistance and directly influences the length n of the hash value. An adversary Eve generates messages $m_1, m_2, m_3, ...$ For each message m_i, she stores the hash value $h(m_i)$ and compares it with the previously generated hash values. If for some $i > j$, $h(m_i) = h(m_j)$, the adversary found a collision (m_i, m_j). The messages should be generated by a deterministic pseudorandom generator, making it possible to reconstruct the messages and store them obsolete. Usually, Eve can expect to find collisions after generating $2^{n/2}$ messages.

3.4.3 Location privacy attacks

The following attacks are directly linked to the target (client) positions. Location information can be correlated with publicly available data or with LBS data. Some of the terms are coined by Ruppel et al. [55].

The known whereabouts attack In the *known whereabouts attack (KWA)*, the attacker has information about the target, e.g., the target's home and work addresses [55] [82]. This can include knowledge about the person's identity or inferred information from another data source. The Strava data analyses [104], which revealed military sides' locations, were made possible because some military personnel used the service. The correlation of satellite images, user names, and ids disclosed the relationship between

[2]The name is inspired by the birthday paradox. Within a group of 23 people, the chance that two of them are born on the same day of the year is higher than 50%. As with 23 people, there are 253 pair combinations possible, which is greater than 365 divided by 2. 97% probability is reached with just 50 people.

people, places, and countries. In this case, the service unintentionally acted maliciously with a non-malicious data scientist using provided raw data.

To avoid a KWA, Strava introduced the possibility to hide private places with circular geofences, see Section 1.3. Location data within the geofences is not recorded or shared. From a sportswoman's perspective, this is easily set up, but the work of Nick Mueller showed that this is insufficient to hide private addresses from being disclosed [16].

The campus attack In a *campus attack (CA)*, an adversary uses locations visited by large amounts of the public, e.g., a university campus, to find target user connections (groups). Solutions that use coordinate transformation methods are vulnerable to this attack. Solutions for privacy-preserving distance measuring can be open to this attack, as the LBS can correlate locations, times, and groups. In this case, an attacker might get to know certain information from a target group, such as a shared secret key. The effectiveness of a CA is highly dependent on the size of the target group. A CA seems not beneficial for small or medium-sized groups, but for larger communities, such as the Strava example, it can be efficient [55] [82].

The stationary user attack In the *stationary user attack (SUA)*, an attacker tries to gain information based on the time targets dwell at locations such as restaurants, workplaces, or cinemas. In this case, a single target position update, maybe related to a user group, could reveal additional information. The SUA can be used by an attacker within methods using time-depended transformations to secure privacy, see Section 5.3.1. In some scenarios, the SUA can be avoided by not producing position updates when the target user is stationary [55].

The map-based attack and mobility pattern attack Using a privacy method based on coordinate transformation, the *map-based attack (MBA)* relates a *transformed* position trace to map structures, such as motorways, tracks, or cycle paths. If an attacker knows about a target's typical routes, the transformed trace could be correlated with road or public transport structures. In this case, the attack is also called *mobility pattern attack (MPA)* [55].

3.4.4 Location security attacks

The usage of location information in access control contexts differs from the privacy use cases. Generally speaking, it is quite the opposite usage. In a privacy context, the target wants to hide the LBS position in a security context, i.e., in a *location-based access control (LBAC)* scenario, the target seeks to prove its location to a service.

Therefore different types of attacks need to be addressed by their equivalent countermeasures. As a reminder, the thesis' focus is on privacy-preserving LBS. A *security attacker* uses the following methods to gain an advantage.

Spoofing, cloning, and replay attacks To get access to a service, the malicious person could *spoof the location (location spoofing, LS)* of the target client, i.e., the location information is faked [105]. This can be done for various positioning methods, i.e., GPS, Wi-Fi, Cell-ID, or Bluetooth.

It can also be referred to as a *location cloning attack (LCA),* i.e., the attacker clones the signal from a different beacon to fake her position.

In the case of a *location replay attack (LRA)*, the target device records the beacon's broadcast data and *replays* them when needed. The attack's apparent countermeasure is not to use an identifier more than once. Countermeasures to all attacks are described in Section 3.4.5

Piggybacking and hijacking *Piggybacking* sometimes referred to as *tailgating*, and *hijacking*, uses a signal of a beacon belonging to service *A* and implements it into another application belonging to service *B*. Service *B* will be triggered each time the target user is within *A*.

Signal extending In the *signal extending attack (SEA)*, a rogue *repeater beacon (RB)* is used to extend the signal of the *origin beacon (OB)* to increase the attacker's distance to the OB.

Relay attacks A *relay attack (RA)* is a SEA, but the signal will be transmitted over the Internet. This attack also requires a physical device present at the location of the OB. This could be a SIM-enabled smartphone or a small single-board computer, such as a Raspberry Pi.

In a *mafia attack (MA)* scenario, a *prover*, e.g., a contactless card, and a *verifier*, e.g., a card reader, interact with two malicious parties, i.e., a malicious prover and verifier, intercepting each other's transactions. The attack is better known under the name *man-in-the-middle (MITM)*. This attack is based on the Grandmaster Chess problem described by John H. Conway in 1976. In this game, a malicious and unskilled player (MUP) challenges two grandmasters simultaneously and relays the moves. The MUP either *wins* against one, or the game results in a draw.

The *terrorist attack (TA)* is a variation of the MA. It includes an honest prover collaborating with a malicious person to deceive an honest verifier. Some researchers consider this relay attack an open problem [106], though different countermeasures have been proposed. The most popular countermeasure is the distance-bounding protocol [107] [108]. It aims to counter the attack by detecting the additional delay needed by the adversary to relay a message. The relay attacks are also described by Dürholz et al. [109] and Boureanu et al. [110] in their work on distance-bounding protocols.

3.4.5 Countermeasures

Table 3.3 provides the overview of the following measures that counter the attacks described in Sections 3.4.1 to 3.4.4.

Long keys An effective way to overcome cipher- and plaintext attacks and the increasing computational brute-force capabilities is to use increasing key length in symmetric and asymmetric encryption schemes.

Ephemeral identifiers Unlike location spoofing, a tag generator is not efficient to counter spoofing and piggybacking. However, using the *ephemeral,* i.e., short-lived

Table 3.3: Attacks and their respective countermeasures

Attacks	Countermeasures
Ciphertext-only	Long keys, strong encryption method
Known-plaintext	Long keys, random values, limit access to encryption key
Chosen-plaintext	Long keys, limit access to encryption key, hybrid encryption
Chosen-ciphertext	Long keys, limit access to decryption device
Small-message	Random values
Common-modulus	Randomly chosen prime factors
Low-encryption-exponent	Large encryption exponent
Replay	One-time tokens and passwords, timestamps, session IDs, signatures
Dictionary	Random values, key stretching, key strengthening, long passwords
Brute-force	Random values, key stretching, key strengthening
Lookup tables	Random values, key stretching, key strengthening, long passwords
Length extension	Double hashing (HMAC)
Birthday	Increase hash output length, random values
Known whereabouts	Temporal or spatial cloaking, exclusion, coordinate transformation
Campus	Secret keys for coordinate transformation (for small groups)
Stationary user	Avoid position updates (maybe infrequently PUs)
Map-based/mobility pattern	time-dependent transformation
Spoofing	Location tag generation, ephemeral IDs
Piggybacking	Ephemeral IDs
Signal extending	ID duplication detection
Relay	Delay-based and physical presence detection

identifier, makes any sniffed ID implemented in a third party application useless and stops these attacks [111].

Random values Adding (pseudo)randomness, also known as *salt*, *pepper*, or *nonce* values, to messages m and passwords p increases the likelihood for brute-force and dictionary attacks to fail. The length of such values and the randomness decide if such an attack is feasible or not. Sections 4.4.1 and 5.3.1 discuss this countermeasure.

Hash output length The output length n needs to be chosen accordingly to make it infeasible to store all hash values. Computational advancements and storage capacities demand increasing the size [90].

Double hashing Within an HMAC, the hash function $h()$ is used twice for hashing the key-inner-padded message and once for hashing the key-outer-padded message. Without this double hashing, an adversary Eve could apply the Merkle-Damgård iteration on any

message m'. She can compute an HMAC value v without the need to know the original message m, see [90] for details in the iteration process.

Key stretching, key strengthening, and key derivation The *key stretching* countermeasure will use a password and a salt value and apply the same hash function iterative, for example, a thousand times. In the *key strengthening* approach, the salt is deleted, i.e., not stored with the password. In this case, an attacker is obliged to do a brute-force search for the random value. A *key derivation* function is used to descend a secret key or value from a password, passphrase, or pseudorandom function, sometimes combined with a random salt value. See also Sections 4.4.1 and 5.3.1.

Coordinate transformation In the context of privacy-preserving LBS, a *time-independent coordinate transformation* masks the relation to the known global reference system but keeps relative information [55].

Time-dependent transformation In relation to the coordinate transformation, a *time-dependent transformation* masks the relations of individual position updates of a single target among itself, i.e., it blurs movement patterns. Relative information of different targets or global references is also remaining [55].

ID duplication detection A countermeasure to detect signal-extending attacks is for transmitting and receiving devices to scan the local environment to check if beacons can be seen with identical IDs. For example, the broadcasting ID cannot be detected by its own beacon in the Bluetooth protocol. Therefore rouge devices that try to extend signals can be detected.

Delay detection The apparent countermeasure in distance bounding protocols is to detect *any difference* between the sending and receiving parties [109] [110]. The adversary in a *man-in-the-middle* attack would be considered to alter given message roundtrip times.

Physical-presence detection The presence of a person can be verified by the necessity of interactions, such as switching a switch, entering a PIN into a keypad, or scanning the biometric attributes of a user. Distance bounding protocols are used in security contexts, which explains their location leakage potentials in privacy use cases [112].

3.5 Chapter summary

The chapter started with defining the terms, roles, and actors. It introduced security and privacy models, highlighting the differences in their objectives. The chapter concluded with an overview of attacks and countermeasures to be considered in the different cryptographic approaches.

E.T. phone home.

E.T. The Extra-Terrestrial (1982)

4 Related Work

This chapter discusses geo encodings, location privacy classifications, and cryptographic schemes, and privacy protocols. The chapter starts with defining polygon shapes and presents two algorithmic methods for a point-in-polygon evaluation with roots in computational geometry.

4.1 Polygons

Different categories of polygon shapes have to be defined to understand point-in-polygon (PIP) algorithms in computational geometry and geographic information systems (GIS). Figure 4.1 shows different polygon classes based on their (mathematical) characteristics, adapted from [113].

The polygon classes can be categorized based on their convexity and non-convexity [114] [115].

- **Convex**: All its interior angles are less than 180°.
- **Simple**: The boundary of the polygon does not cross itself. All convex polygons are simple.
- **Complex or self-intersecting**: The boundary of the polygon crosses itself.
- **Concave**: At least one interior angle is greater than 180 degrees. The polygon is non-convex and simple.
- **Star-shaped**: The whole interior is visible from at least one point, without crossing any edge. The polygon must be simple and convex or concave. All convex polygons are star-shaped.
- **Star polygon**: A polygon that self-intersects in a regular way. A polygon cannot be both a star and star-shaped.

Additionally, polygons can be categorized based on equality and symmetry features.

- **Equiangular**: All angles are equal.
- **Equilateral**: All edges have the same length.

- **Regular**: A polygon that is both equilateral and equiangular.
- **Cyclic**: All corners lie on a single circle, i.e., the circumcircle.
- **Tangential**: All sides are tangent to an inscribed circle, i.e., a circle that lies inside the polygon. An inscribed circle can be drawn inside the regular convex example.
- **Isogonal or vertex-transitive**: All corners lie within the same symmetry orbit. The polygon is also cyclic and equiangular.
- **Isotoxal or edge-transitive**: All sides lie within the same symmetry orbit. The polygon is also equilateral and tangential.

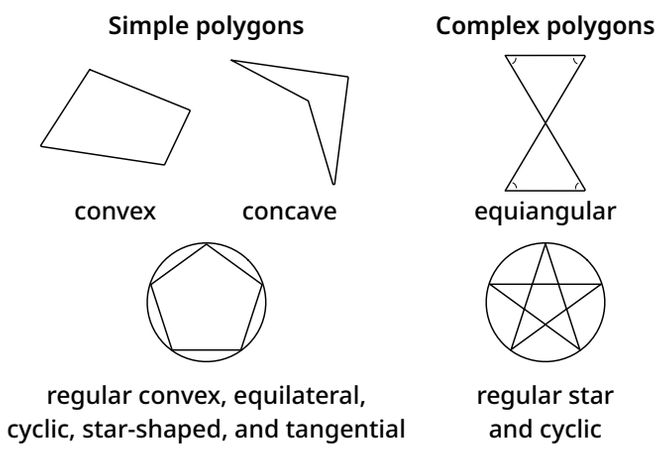


Figure 4.1: Examples of simple and complex polygons in different categories.

The arbitrary polygons used throughout the thesis are mostly *simple convex* and *concave* shapes. *Complex* or *self-intersecting* polygons can also be considered multiple separate polygons, with vertices meeting in one or more points, i.e., coordinates.

4.1.1 Point-in-polygon algorithms

To evaluate if a point is within an arbitrary-shaped *simple* polygon, a point-in-polygon (PIP) algorithm is needed.

4.1.1.1 Ray casting algorithm

One of the most straightforward algorithms for simple polygons, i.e., non-crossing lines and no wholes, is counting how many intersections a ray passes from the point through the polygon. It is also called the *crossing number algorithm* or the *even-odd rule algorithm*. If the point is outside the polygon, then the ray crosses the polygon boundary in an *even* number of times. If the point is inside the polygon, the ray passes the point an *odd* number of times, see Figure 4.2, adapted from [116].

Pseudocode for this algorithm is shown in Listings 4.1 and 4.2 below, as presented by Erickson in [117]. The publication proves the algorithm using the Jordan Curve Theorem.

```
PointInPolygon(P[1,...,n], q)
    sign = −1
    P[0] = P[n] //to have a closed polygon
    for i = 0 to n − 1
        sign = sign * RightCross(q, P[i], P[i + 1])
        if sign = 0 aboard loop
    return sign
```

Listing 4.1: Pseudocode for the point in polygon crossing number algorithm.

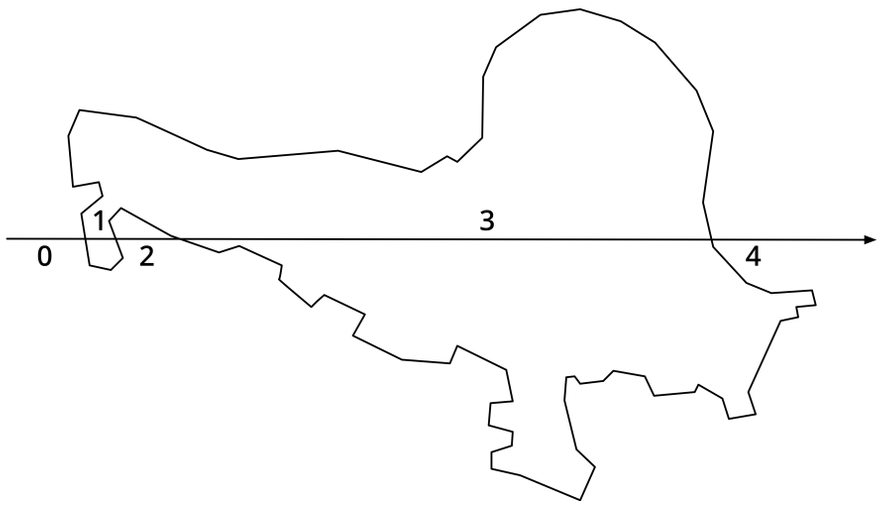


Figure 4.2: Example of the crossing number algorithm.

```
RightCross(q,r,s)
    if q.y = r.y = s.y
      if r.x <= q.x <= s.x or s.x <= q.x <= r.x
        return 0
      else
        return +1
    if q.y = r.y und q.x = r.x
      return 0
    if r.y > s.y
      swap r and s
    if q.y <= r.y or q.y > s.y
      return +1
    set delta = (r.-xq.x) * (s.-yq.y) - (r.-yq.y) * (s.-xq.x)
    if delta > 0
      return -1
    else if delta < 0
      return +1
    else
      return 0
```

Listing 4.2: Pseudocode for the subroutine testing if a ray crosses the polygon to its right side.

The input polygon *P[1,...,n]* is represented by an array of consecutive and distinct vertices. The `PointInPolygon()` algorithm returns +1, -1, or 0 to indicate that the query point *q* lies inside, outside, or directly on *P*, respectively. The `RightCross()` submethod treats any polygon vertex that lies on the ray as though it were slightly above. This trick automatically takes care of limiting cases. The algorithm complexity is *O(n)* as the number of calculations is *only* dependent on the number of vertices *n*.

4.1.1.2 Winding number algorithm

An alternative to the previous approach is presented in Hormann et al. [118] and is based on the point's *winding number*. The winding number of a closed polygon in the plane of a given point is an integer representing the total number of times that curve travels counterclockwise around the point, see Figure 4.3 [119].

If the winding number is non-zero, the point is inside the polygon. Therefore the algorithm is called *non-zero rule* or *non-zero winding rule*. There are algorithms based on

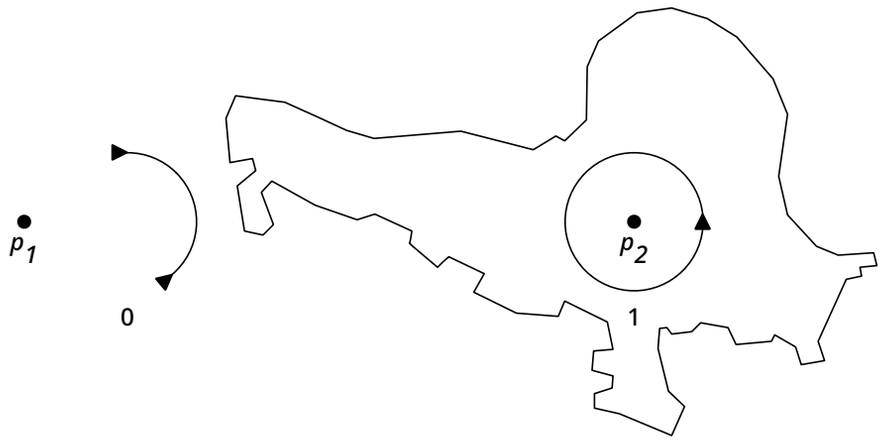


Figure 4.3: Two different winding number examples, with p_1 having winding number 0, which means that this point is outside the polygon and with p_2 having winding number 1, i.e.,$\neq 0$, which means that that point is inside the polygon.

summing up angles or on signed crossings (left-to-right or right-to-left).

Other point location approaches include *slab decomposition*, *monotone subdivisions*, *triangulation refinement*, and *trapezoidal decomposition* [114].

The following section will introduce how geo encoding systems based on grid cells can represent arbitrarily shaped polygons and points later used within cryptographic schemes for a privacy-preserving PIP evaluation.

4.2 Geo encodings

A geofence and a target position need a representation, i.e., a location encoding system, also known as a *geo encoding* or *reference system*. It needs to fulfill the LBS class objectives, including the resolution of the geofence areas, the possibility to represent any arbitrary shape, and the calculation and communication costs. The geo encoding also needs to adhere to follow the privacy-preserving approaches.

The most common representation is the longitude and latitude coordinate system of the World Geodetic System in its revision from 1984 (WGS 84) [2]. The examples below show that most other systems encode these coordinates into a single code, such as the Geohash representation [120]. As with similar systems, Geohash partitions the Earth into tiles, and these encodings represent specific areas. Location encoding systems are designed with various general requirements in mind, such as *ease of memorization*, *avoidance of similar characters*, *multiple resolutions with various numbers of digits*, *comparability*, *offline de- and encoding*, *static representations*, or *non-licensed usage* [121]. As this thesis is about privacy-preserving geofences, space partitioning focuses on discrete global grids (DGG), specifically *hierarchical* reference systems. Other properties to classify a DGG are tile shape and granularity, i.e., grid resolution or cell size. Tile-shapes can be *regular*, *semi-regular*, or *irregular*. The granularity or grid resolution of a hierarchical DGG defines the average area of a cell.

Traditionally, cartography uses *map projections* where points on the Earth's surface are mapped to the points on a planar map. This causes distortions, for example, visible in the Mercator projection and discontinuities along the meridian. Besides, this type of projection is unable to represent the poles. Location encoding systems include the

following: WGS84 [2], Geohash [120], Mapcode [122], Open Post Code [123], Natural Area Code [124], Maidenhead Locator System (MLS), Universal Transverse Mercator coordinate system [125], Open Location Code/Plus Codes [126], Hierarchical Equal Area isoLatitude Pixelization (HEALPix) [127], what3words [128], and the S2 grid cell approach [129].

Some location encoding systems characteristics are described below, alongside examples for longitude/latitude, Geohashes, and S2 cells (all for the Berlin area of Germany).

4.2.1 Geographic coordinate system

The standard *geographic coordinate system* belongs to the group of *horizontal position representation* and describes positions on the Earth's globe with the values *longitude* (Long., λ, lambda, representing the x-axis) and *latitude* (Lat., ϕ, phi, representing the y-axis), as shown in Figure 4.4, adapted from [130]. Besides, the *altitude* or *depth* can be added as additional pieces of information to detail the position at a given location on the Earth's surface. The origin, i.e ., zero point (0°0′0′′N 0°0′0′′E), is located about 625km south of Tema, Ghana, in the Gulf of Guinea, also known as Null Island. Some Strava activity events are mapped to this location, as users set these coordinates to disguise their actual location [104].

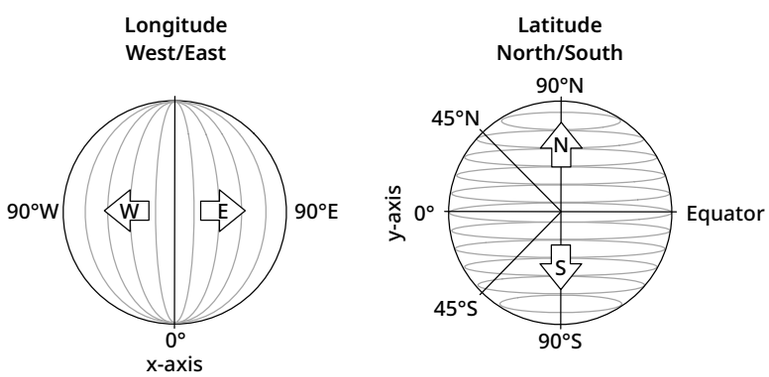


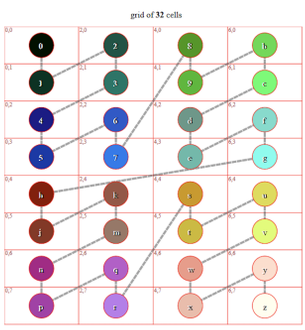
Figure 4.4: The longitudes are perpendicular, and the latitudes are parallel to the equator.

The World Geodetic System (WGS 84) used in cartography, geodesy, and general Global Navigation Satellite Systems (GNSS) defines the coordinate system and derived constants. Long/Lat coordinates are measured in degrees (1°), minutes (1′), and seconds (1′′). Depending on the spherical position, they represent different meter measures. The Ernst-Reuter-Platz in Berlin, Germany, is located at 52°30′45.0′′N 13°19′18.8′′E, north of the equator east of Greenwich. The same point can be represented in its decimal representation, 52.512512, 13.321877. The disadvantages of this representation are the singularities at both poles where the longitude is undefined. The grid is non-linear near the poles, with a discontinuity at the ±180 meridian. Since the system was designed to express point locations on the Earth, it is not suited for addressing areas or memorable postcodes.

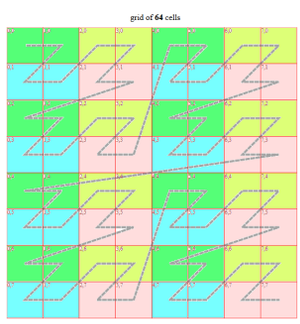
4.2.2 Geohash

The Geohash is a public domain system invented by Gustavo Niemeyer in 2008 [120]. The key characteristic of a Geohash is its use of alphanumeric characters and a z-order space-filling curve. There are 12 precision levels, one for each character. Removing characters

from the end of the code reduces the precision and increases its size. Therefore, a hierarchical spatial data structure subdivides each grid cell into 32 child cells. Grid cells with similar prefixes are mostly closer together. The opposite, however, is not guaranteed. Geohashes have been used to represent Geofences [131]. A Geohash example is u33db, see Figure 4.5.



(a) Odd digits are represented by a И-order curve. Latitude and longitude values of the decoded pair have different uncertainties.



(b) Even digits are represented by a Z-order curve in a regular grid where a decoded pair (latitude, longitude) has uniform uncertainty.

(c) Geohash examples with level of four characters

Figure 4.5: Geohash curve representation with different grid cell levels in (a) 32 cells and (b) 64 cells). (c) shows example Geohash areas with an encoding level of four characters.[1]

Table 4.1: Example of how to encode latitudes into a Geohash

# Bit	Latitude min	Latitude mid	Latitude max	Bit result
1	-90	0	90	1
2	0	45	90	1
3	45	67.5	90	0
4	45	56.25	67.5	0
5	45	50.625	56.25	1
6	50.625	53.4375	56.25	0
⋮	⋮	⋮	⋮	⋮

Geohash-based geofences are suited for the approaches discussed in Faber [132], namely privacy-preserving substring matching to address Objective 6, in which the service should be able to limit the geofence size without getting to know its location. The Geohash alphabet (32ghs) uses digits *0-9* and all lower case letters except *a*, *i*, *l*, and *o*, resulting in 32 different values, i.e., a base32 representation, see Table 4.2. Some variants, such as the Geohash-int [133] from 2014, using 64-bit values, and the Geohash-Hilbert [134] version, using a Hilbert space-filling curve from 2016. The last variant is similar to the S2 system described in the following section. The variants are used primarily in non-human-readable use cases. There is also a Java implementation provided by Silvio Heuberger [135].

Using the alphabet 32ghs and the constant value of $B = 32$, the example Geohash u33db is converted into a decimal value as follows.

[1] Image sources: (a) and (b) by P.P. Kraus under the CC BY-SA 4.0 license, (c) The underlying map data is provided by OpenStreetMap (OSM) under the Open Database License (ODbL), openstreetmap.org/copyright.

Table 4.2: Geohash alphabet

Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B32	0	1	2	3	4	5	6	7	8	9	b	c	d	e	f	g
Dec	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
B32	h	j	k	m	n	p	q	r	s	t	u	v	w	x	y	z

$$[\mathrm{u33db}]_{32\mathrm{ghs}} = [(\mathrm{u} \times B^4) + (3 \times B^3) + (3 \times B^2) + (\mathrm{d} \times B^1) + (\mathrm{b} \times B^0)]_{32\mathrm{ghs}} =$$

$$[26]_{10} \times B^4 + [3]_{10} \times B^3 + [3]_{10} \times B^2 + [12]_{10} \times B^1 + [10]_{10} \times B^0 = 27364746$$

The binary representation of the example value u33db is as follows.

$$[\mathrm{u}]_{32\mathrm{ghs}} = [26]_{10} = [11010]_2$$

$$[3]_{32\mathrm{ghs}} = [3]_{10} = [00011]_2$$

$$[3]_{32\mathrm{ghs}} = [3]_{10} = [00011]_2$$

$$[\mathrm{d}]_{32\mathrm{ghs}} = [12]_{10} = [01100]_2$$

$$[\mathrm{b}]_{32\mathrm{ghs}} = [10]_{10} = [01010]_2$$

All example results are in the following table.

Table 4.3: Geohash example representations

Representation	Values				
Base32	u	3	3	d	b
Decimal	26	3	3	12	10
Binary	11010	00011	00011	01100	01010
Longitude	1-0-0	-0-1-	0-0-1	-1-0-	0-0-0
Latitude	-1-1-	0-0-1	-0-1-	0-1-0	-1-1-

Table 4.4: Geohash cell sizes

Geohash length	Cell width (≤)	Cell height (≤)
1	5000km	5000km
2	1250km	625km
3	156km	156km
4	39.1km	19.5km
5	4.89km	4.89km
6	1.22km	0.61km
7	153m	153m
8	38.2m	19.1m
9	4.77m	4.77m
10	1.19m	0.596m
11	149mm	149mm
12	37.2mm	18.6mm

4.2.3 S2

S2 is a library for spherical geometry that provides robustness, flexibility, and performance as other planar geometry libraries [129]. The name S2 is derived from the mathematical notation of the unit sphere S^2. The S2 C++ open source library has been written primarily by Eric Veach and published under the Apache License 2. Other significant contributors include Jesse Rosenstock, Eric Engle (Java port lead), Robert Snedegar (Go port lead), Julien Basch, and Tom Manshreck [136].

With the S2 approach of using *spherical projection*, the Earth's surface is projected to a perfect mathematical sphere, limiting the distortion through projection to a maximum of 0.56%. S2 preserves the correct topology of the Earth, and there are no singularities or discontinuities [129].

Using spherical geometry rather than an ellipsoid that would be closer to the natural Earth sphere is due to performance and robustness requirements.

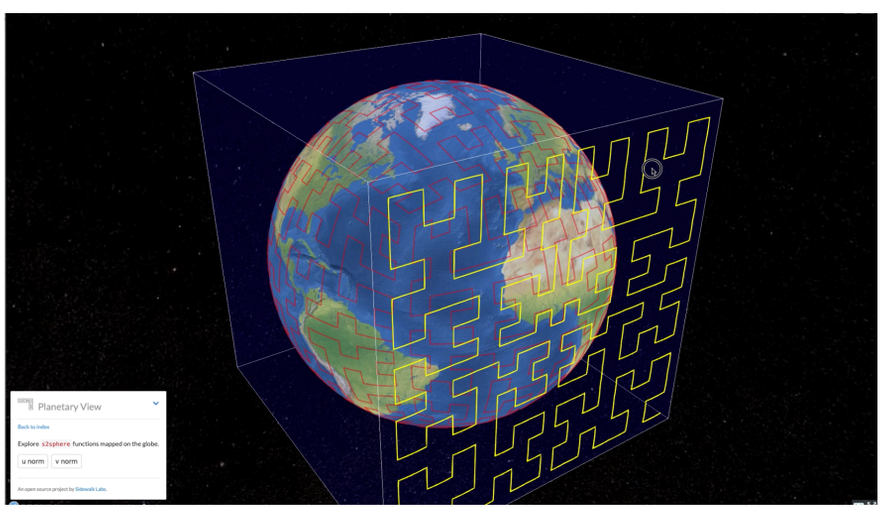


Figure 4.6: The Hilbert curve on the S2 cube is projected onto the globe. A distortion of the 3D representation can be seen, showing the square faces of the ideal cube as different shapes on the sphere.[2]

The S2 library is designed to make the core operations 100% robust. This means that the outputs of each operation are mathematically strict for all input values. Calculating the intersection of two polygons is guaranteed to be topologically correct up to the creation of degeneracies. However, it is also ensured that the output boundary stays within a user-specified tolerance of an accurate, mathematically exact result.

Robustness is also essential for using the S2 library for arbitrarily shaped geofences, as unexpected results from a very low-level function may cause higher-level usage errors. S2 attempts to be *precise* regarding *mathematical definitions* and *numerical accuracy*. According to the author's website [129], S2 achieves this goal through *a combination of techniques from computational geometry, including conservative error bounds*, which measure the error in certain calculations, *exact geometric predicates*, which determine the true mathematical relationship among geometric objects, and *snap rounding*, a technique that allows rounding results to finite precision without creating topological errors.

[2]Image source: Screenshot of the http://s2.sidewalklabs.com/planetaryview/ website. Sidewalk Labs is an urban planning and infrastructure subsidiary of Google. Google founded it in 2015 and became an Alphabet company in 2016.

In addition, S2 is a very modular library and enables the development of arbitrary geometry subtypes and data storage in multiple file formats. The methods allow control over semantics, i.e., geometries with open, semi-open, or closed boundaries.

The performance is excellent due to an in-memory edge index data structure. The selection speed of the polygons containing the point of interest is in the range of nanoseconds. The S2 library has output-depended run times, e.g., an intersection near the German Island Sylt depends on this specific area rather than the overall complexity of Germany.

S2 provides the following properties and features [129].

- Representations of angles, intervals, latitude-longitude points, unit vectors, and various operations on these types.
- Geometric shapes over the unit sphere, such as spherical caps (“discs”), latitude-longitude rectangles, polylines, and polygons.
- Robust, constructive operations (e.g., union) and boolean predicates (e.g., containment) for arbitrary collections of points, polylines, and polygons.
- Fast in-memory indexing of collections of points, polylines, and polygons.
- Algorithms for measuring distances and finding nearby objects.
- Robust algorithms for snapping and simplifying geometry (with accuracy and topology guarantees).
- A collection of efficient yet exact mathematical predicates for testing relationships among geometric objects.
- Support for spatial indexing, including the ability to approximate regions as collections of discrete S2 cells.

Nevertheless, the following objectives are not within the library’s scope.

- Planar geometry.
- Conversions to/from standard GIS formats.

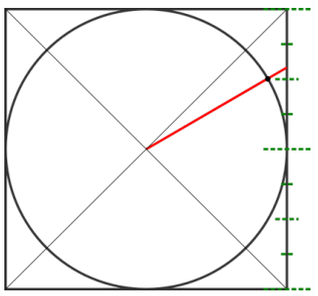
Next to the native C++ implementation, the S2 library functionality is partially available in Go, Java, and Python.

The S2 library uses the concept of *cells*. These cells are a hierarchical decomposition of the Earth’s sphere into compact representations of regions and points. Regions can be approximated using the same cell concept. Each region features compact 64-bit integers. They have the following features.

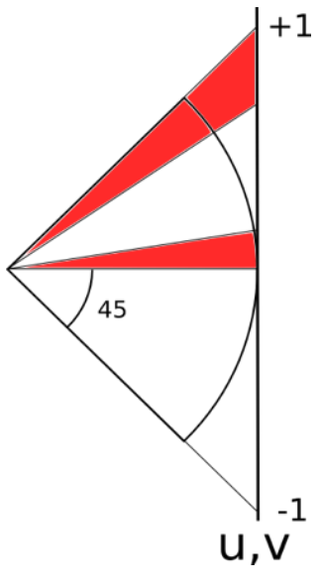
- They have a resolution, e.g., to be used for geographic features.
- They are hierarchical.
- They have levels representing areas.
- Areas with similar levels have similar surface measures.
- The containment query for arbitrary regions is very fast.

Hence, these features make the S2 library a good tool for constructing and evaluating privacy-preserving geofence evaluation in this thesis. The S2 concept description can be found in Christian S. Perone’s blog post [137]. It also refers to a Google presentation by Octavian Procopiuc [138].

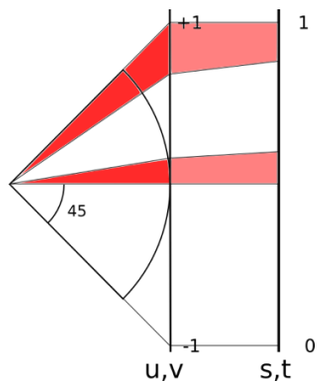
The Earth’s globe is represented by a sphere. This sphere is projected into a cube, see Figure 4.6.



(a) 2D representation of a beam going from the Earth center (circle) onto the S2 cube (square), where it will follow a Hilbert curve.



(b) The initial u,v values beamed onto the square would represent areas on the circle covering the same space differently on the square in terms of m^2.



(c) To solve this distortion inequality the s,t values are introduced to reduce the space differences of the Earth areas in relation to the cube area representation in S2.

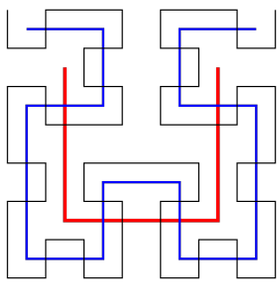
Figure 4.7: The three figures show the S2 process for reducing the area differences between the globe and its cube representation by introducing the s,t values after the sphere-centered rays hit the u,v values on the idealized cube representation.[3]

The cube has the coordinates $[-1,1] \times [-1,1] \times [-1,1]$. The projection p=(lat, long) can be described as a vector (x,y,z), see Figure 4.7a. A quadtree is used on each of the six cube faces. In the first step, every projection p is represented by a quadtree cell. In the second step, the coordinates (x,y,z) are transformed into the values (face,u,v). But with this conversion, same area cells on the cube have different sizes on the sphere, see Figure 4.7b. The highest to lowest area ratio would be 5.2. In the third step, a non-linear transformation is used, (face, u, v) becomes (face, s, t) with a range of $[0,1]$, see Figure 4.7c.

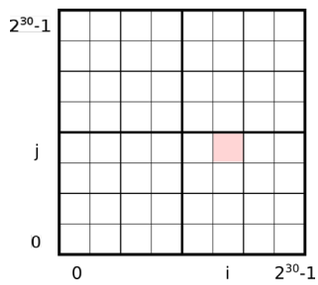
Even (face, s, t) is discretized to (face, i, j), shown with the quadtree cells' most significant bits of i an j in Figure 4.8b. (face, i, j) is called *S2 cell-ID*, which is derived from an enumeration of a 1-dimensional Hilbert space-filling curve, see Figure 4.8a. It preserves spatial locality, close 2-dimensional cells on the cube are also close in the 1-dimensional enumeration, see Figure 4.9 [137].

A 64-bit Integer number represents each cell. The first three bits represent one of the six faces of a cube $[0,5]$ (as in Figure 4.10), the following numbers describe the position along the 1D Hilbert curve $[0, 2^{30}-1] \times [0, 2^{30}-1]$, and the least significant bit (LSB) followed by 0s indicates the cell level. The cell level ranges from 0 (largest area, 1/6 of the Earth, i.e., one side of the cube) to 30 (smallest area, around 1 cm^2), also referred to as an *S2 point*.

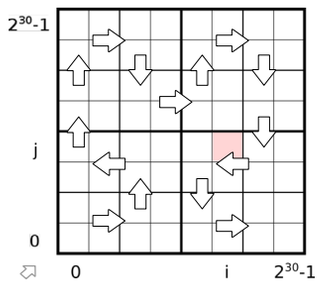
[3]Image source: Figures are published on the original S2 website https://s2geometry.io/.
[4]Image source: Figures are of the original S2 website https://s2geometry.io/ by s2geometry-io@google-groups.com.
[5]Image source: Images used in blog post, Google's S2, geometry on the sphere, cells, and Hilbert curve, by



(a) An example Hilbert curve showing three different levels, low levels in red (big area), middle level in blue, and high level in black (small area).

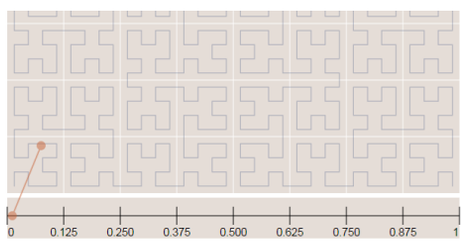


(b) S2 raster consists of quadtree cells, the most significant bits of i and j (e.g., grey area), and one of six faces. The discretized (s,t) values, (i,j) range from 0 to $2^{30}-1$ on the x- and y-axis.

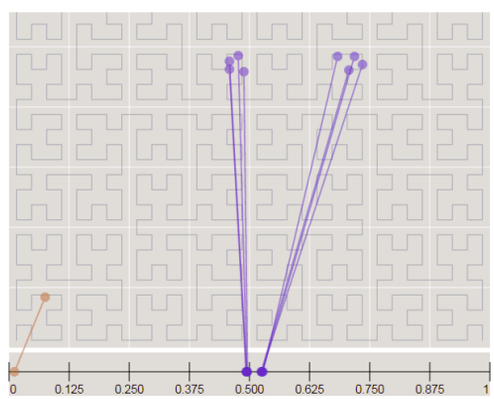


(c) The Hilbert curve on top of the S2 raster. Starting point of the curve is position is $0, 0$ on the bottom left. Endpoint of the curve is in the bottom right corner.

Figure 4.8: S2 Hilbert space-filling curve and raster examples.[4]



(a) This is an example of a point in a 2-dimensional space on the Hilbert curve represented with a 1-dimensional value on the x-axis at just above 0.



(b) Locality on the sphere and in S2. The four points of each cluster (group) close on the Earth's surface are also close on the 1-dimensional x-axis. One group being below 0.5, the other larger than 0.5

Figure 4.9: The idea behind the S2 representation is that two points or areas on the Earth's surface that are close to each other should also be close to each other in their S2 encoding. This means that in their numerical representation, these two points (or areas) are close to each other in S2. In (b), the two groups of four closely spaced points are similarly close in their x-axis representation. It is important to note that the conversion is not true. Two S2 points on the 1-dimensional axis do not necessarily have to be close to each other on the Earth's surface.[5]

The smallest S2 cells are the leaves of the quadtree representation. Levels 0 to 29 are the parents of the leave cells. The following binary number is an S2 cell-ID example.

010 10010100 01000101 00011001 01000100 01010001 10010100 10000000 00000

This cell is located in face two (010) with a cell level of 24, indicated by the LSB at position 13. Each square centimeter on Earth can be represented utilizing this structured 64-bit integer number. Figure 4.10 shows the six top-level face cells [129], including their values 0-5. The start and endpoint of the Hilbert curve are marked with red arrows. The starting face is cell 0 on the bottom left. The final face is cell five on the top right. The source points out that the geographic representation is not completely accurate, but the figure provides an overview of which areas belong to which face [129].

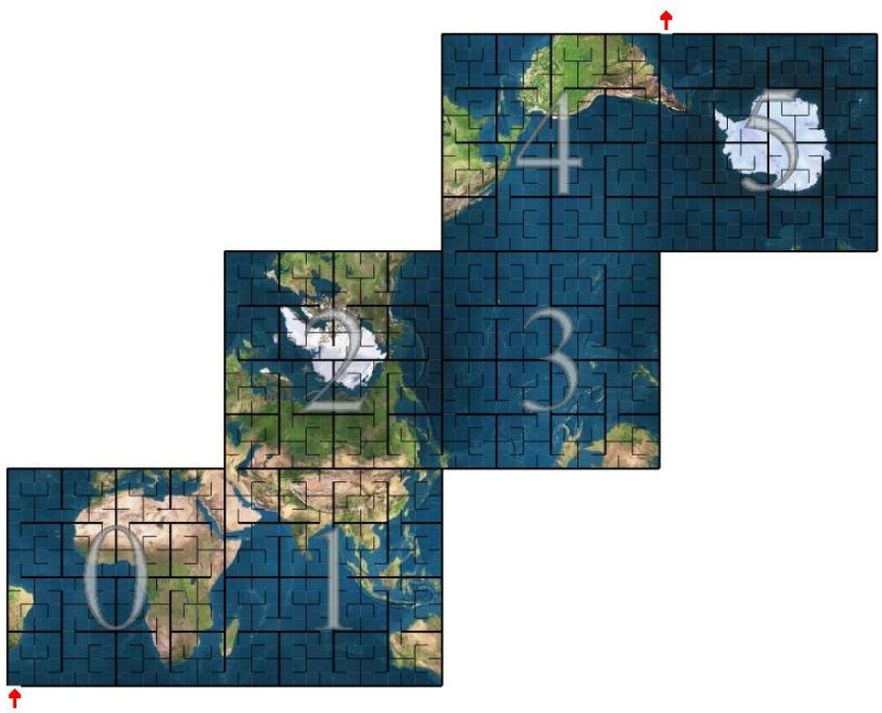


Figure 4.10: The six faces of the S2 Earth cube in a flat position. The six faces are represented by their binary code (000-101) at the leftmost bits in an S2 cell. The Hilbert curve covering all six faces begins at face 0 in the lower-left corner, indicated by the red arrow. The Hilbert curve ends its space-filling function at surface 5, also indicated by a red arrow.[6]

Figure 4.11 shows the shape of S2 cells around different areas of the Earth. To generate the images, the online Sidewalk Labs region coverer service was used [139]. The shapes differ due to the spherical projection. Not having a uniform shape is the only disadvantage for some approaches presented in Chapter 5. The alternative geo encoding used in those cases is the Geohash representation described above. Geohash replaces, in such cases, the geo encoding as an alternative. A comparison is shown in the evaluation chapter.

Christian S. Perone, https://blog.christianperone.com/2015/08/googles-s2-geometry-on-the-sphere-cells-and-hilbert-curve/ under the CC BY-NC 4.0 (2015).

[6]Image source: Figure is published on the original S2 website https://s2geometry.io/.

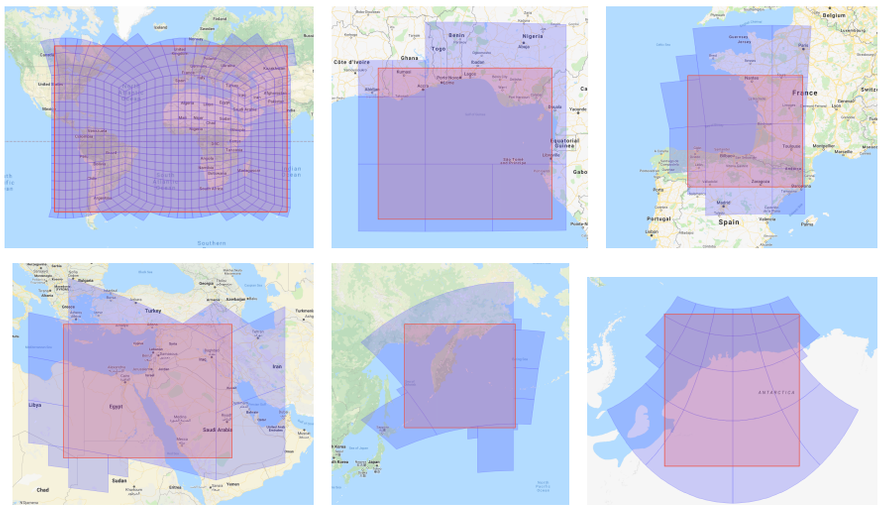


Figure 4.11: Shown are rectangular geofences and their S2 grid cells with different planes on different sides of the Earth cube. The red areas show the geofence and the blue areas represent the S2 grid cell representation. Large areas are used to show the distortion of the S2 cells resulting from the transformation from the sphere to the cube and the S2 representation covering multiple surfaces, e.g., in the figure below left. The S2 online region coverer service from Sidewalk Labs was used to create the examples.[7]

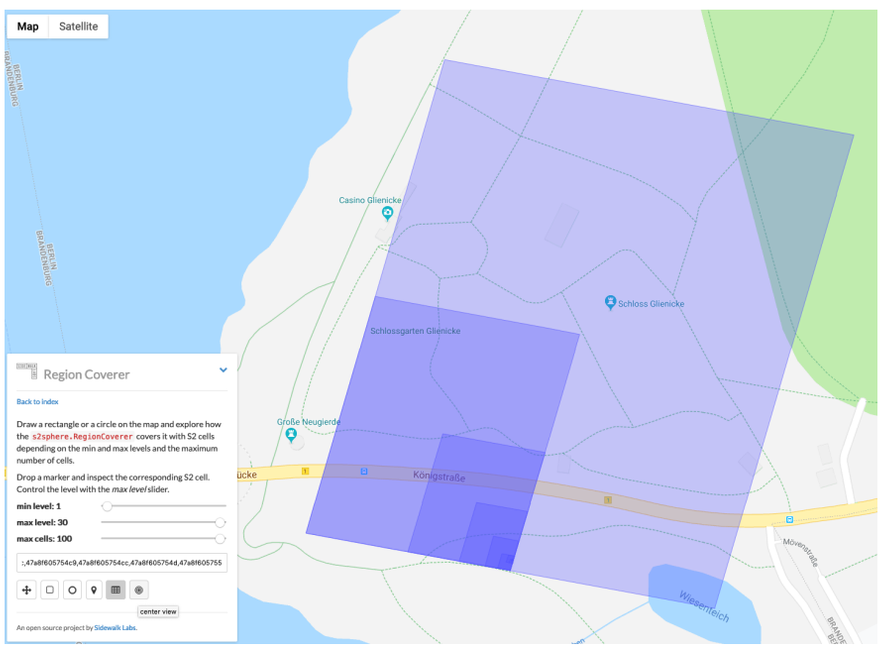


Figure 4.12: Example of S2 grid cells with a subordinate face (smallest quadrilateral) and its parent faces (subordinate larger faces). Each of the parent faces (lower level) includes each previous child face (higher level).[8]

$$\sum_{n=0}^{30} 6 * 4^n = 9 * 10^{18} \neq 2^{64} \tag{4.1}$$

Table 4.5 shows the S2 cell statistics, including example area values and the total number

[7] Image sources: Screenshots of the http://s2.sidewalklabs.com/regioncoverer/ website. Underlying map data by Map data ©2018, Inst. Geogr. Nacional, AfriGIS (Pty) Ltd, Inst. Geogr. Nacional, GeoBasis-DE/BKG (©2009), Mapa GISrael, TMap Mobility, Google.

[8] Image source: Screenshot of the http://s2.sidewalklabs.com/regioncoverer/ website. Underlying map data by Map data ©2018 GeoBasis-DE/BKG (©2009), Google.

of cells for each level [129]. Different cell levels describe the accuracy of the geofence in this thesis. The LSB shifts two places to the left for each parent level. Due to this construction, the amount of S2 cells in total differs from the values that a 64-bit integer number can represent, as seen in Equation 4.1. Tables 4.6 to 4.8 show various S2 cell representations, i.e., *binary*, *integer*, *hex*, and *token* (hex without zeros).

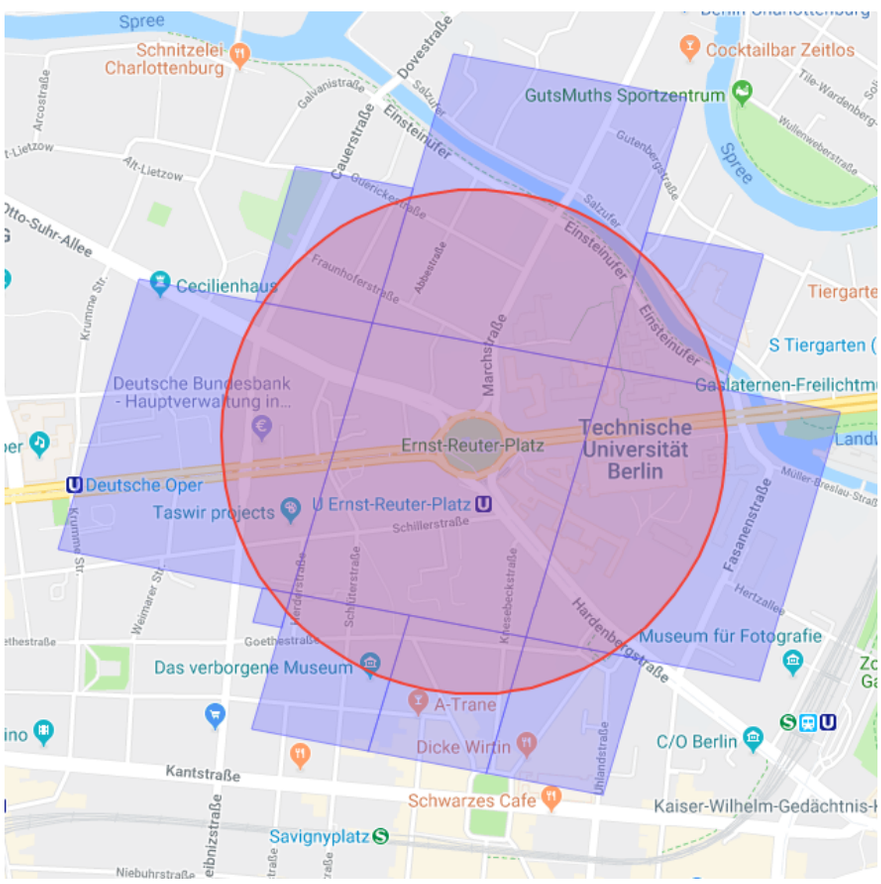


Figure 4.13: A representation of one circular geofence (red) with ten S2 grid cells (blue) using three different levels.[9]

Table 4.10 summarizes the characteristics of the relevant geo encoding systems used throughout the thesis.

4.3 Location privacy

In 2011, Andersen and Kjærgaard [140] categorized location privacy methods for pervasive computing use cases. They define five categories of privacy methods *anonymity*, *classical security*, *spatial obfuscation*, *temporal obfuscation*, and *protocol*. Classical cryptography is subdivided into *cryptography* and *security policies*. The term method protocol is defined as a "... custom protocol to enhance privacy, usually combining anonymity and spatial obfuscation". They also define categories of LBS and their properties. The LBS categories are *point of interest (POI)*, *social networking (SN)*, *collaborative sensing (CS)*, and *route tracing (RT)*. Their properties are subcategorized as *receivers*, *number of receivers*, *number of users*, *identity importance*, *spatial sensitivity*, *temporal sensitivity*,

[9]Image source: Screenshot of the http://s2.sidewalklabs.com/regioncoverer/ website. Underlying map data by Map data ©2018 GeoBasis-DE/BKG (©2009), Google.

Table 4.5: S2 cell statistic

Level	Min. area	Max. area	Avg. area	Units	Min. edge (UK)	Max. edge (UK)	Min. edge (US)	Max. edge (US)	No. of cells
0	85011012.19	85011012.19	85011012.19	km^2	7842 km	7842 km	7842 km	7842 km	6
1	21252753.05	21252753.05	21252753.05	km^2	3921 km	5004 km	3921 km	5004 km	24
2	4919708.23	6026521.16	5313188.26	km^2	1825 km	2489 km	1825 km	2489 km	96
3	1055377.48	1646455.50	1328297.07	km^2	840 km	1167 km	1130 km	1310 km	384
4	231564.06	413918.15	332074.27	km^2	432 km	609 km	579 km	636 km	1536
5	53798.67	104297.91	83018.57	km^2	210 km	298 km	287 km	315 km	6,000
6	12948.81	26113.30	20754.64	km^2	108 km	151 km	143 km	156 km	24,000
7	3175.44	6529.09	5188.66	km^2	54 km	76 km	72 km	78 km	98,000
8	786.20	1632.45	1297.17	km^2	27 km	38 km	36 km	39 km	393,000
9	195.59	408.12	324.29	km^2	14 km	19 km	18 km	20 km	1,573,000
10	48.78	102.03	81.07	km^2	7 km	9 km	9 km	10 km	$6 \cdot 10^6$
11	12.18	25.51	20.27	km^2	3 km	5 km	4 km	5 km	$25 \cdot 10^6$
12	3.04	6.38	5.07	km^2	1699 m	2 km	2 km	2 km	$100 \cdot 10^6$
13	0.76	1.56	1.27	km^2	850 m	1185 m	1123 m	1225 m	$402 \cdot 10^6$
14	0.19	0.40	0.32	km^2	425 m	593 m	562 m	613 m	$1610 \cdot 10^6$
15	47520.30	99638.93	79172.67	m^2	212 m	296 m	281 m	306 m	$6 \cdot 10^9$
16	11880.08	24909.73	19793.17	m^2	106 m	148 m	140 m	153 m	$25 \cdot 10^9$
17	2970.02	6227.43	4948.29	m^2	53 m	74 m	70 m	77 m	$103 \cdot 10^9$
18	742.50	1556.86	1237.07	m^2	27 m	37 m	35 m	38 m	$412 \cdot 10^9$
19	185.63	389.21	309.27	m^2	13 m	19 m	18 m	19 m	$1649 \cdot 10^9$
20	46.41	97.30	77.32	m^2	7 m	9 m	9 m	10 m	$7 \cdot 10^{12}$
21	11.60	24.33	19.33	m^2	3 m	5 m	4 m	5 m	$26 \cdot 10^{12}$
22	2.90	6.08	4.83	m^2	166 cm	2 m	2 m	2 m	$105 \cdot 10^{12}$
23	0.73	1.52	1.21	m^2	83 cm	116 cm	110 cm	120 cm	$422 \cdot 10^{12}$
24	0.18	0.38	0.30	m^2	41 cm	58 cm	55 cm	60 cm	$1689 \cdot 10^{12}$
25	453.19	950.23	755.05	cm^2	21 cm	29 cm	27 cm	30 cm	$7 \cdot 10^{15}$
26	113.30	237.56	188.76	cm^2	10 cm	14 cm	14 cm	15 cm	$27 \cdot 10^{15}$
27	28.32	59.39	47.19	cm^2	5 cm	7 cm	7 cm	7 cm	$108 \cdot 10^{15}$
28	7.08	14.85	11.80	cm^2	2 cm	4 cm	3 cm	4 cm	$432 \cdot 10^{15}$
29	1.77	3.71	2.95	cm^2	12 mm	18 mm	17 mm	18 mm	$1729 \cdot 10^{15}$
30	0.44	0.93	0.74	cm^2	6 mm	9 mm	8 mm	9 mm	$7 \cdot 10^{18}$

Table 4.6: Example of S2 cell levels and binary values

Level	Binary
0	01010000000000000000000000000000000000000000000000000000000000000000
1	01000100000000000000000000000000000000000000000000000000000000000000
2	01000111000000000000000000000000000000000000000000000000000000000000
3	01000111110000000000000000000000000000000000000000000000000000000000
4	01000111101100000000000000000000000000000000000000000000000000000000
5	01000111101011000000000000000000000000000000000000000000000000000000
6	01000111101010010000000000000000000000000000000000000000000000000000
7	01000111101010001100000000000000000000000000000000000000000000000000
8	01000111101010001111000000000000000000000000000000000000000000000000
9	01000111101010001111010000000000000000000000000000000000000000000000
10	01000111101010001111011100000000000000000000000000000000000000000000
11	01000111101010001111011001000000000000000000000000000000000000000000
12	01000111101010001111011000010000000000000000000000000000000000000000
13	01000111101010001111011000001000000000000000000000000000000000000000
14	01000111101010001111011000001010000000000000000000000000000000000000
15	01000111101010001111011000001010100000000000000000000000000000000000
16	01000111101010001111011000001010111000000000000000000000000000000000
17	01000111101010001111011000001010111010000000000000000000000000000000
18	01000111101010001111011000001010111010100000000000000000000000000000
19	01000111101010001111011000001010111010101000000000000000000000000000
20	01000111101010001111011000001010111010101010000000000000000000000000
21	01000111101010001111011000001010111010101001100000000000000000000000
22	01000111101010001111011000001010111010101001101000000000000000000000
23	01000111101010001111011000001010111010101001100110000000000000000000
24	01000111101010001111011000001010111010101001100100100000000000000000
25	01000111101010001111011000001010111010101001100100011000000000000000
26	01000111101010001111011000001010111010101001100100010110000000000000
27	01000111101010001111011000001010111010101001100100010100100000000000
28	01000111101010001111011000001010111010101001100100010100011000000000
29	01000111101010001111011000001010111010101001100100010100010110000000
30	01000111101010001111011000001010111010101001100100010100010111100000

Table 4.7: The equivalent values in integer - hex - token representation

Level	Integer	Hex	Token
0	5764607523034234880	0x5000000000000000	5
1	4899916394579099648	0x4400000000000000	44
2	5116089176692883456	0x4700000000000000	47
3	5170132372221329408	0x47c0000000000000	47c
4	5165628772593958912	0x47b0000000000000	47b
5	5164502872687116288	0x47ac000000000000	47ac
6	5163658447756984320	0x47a9000000000000	47a9
7	5163588079012806656	0x47a8c00000000000	47a8c
8	5163640855570939904	0x47a8f00000000000	47a8f
9	5163645253617451008	0x47a8f40000000000	47a8f4
10	5163648552152334336	0x47a8f70000000000	47a8f7
11	5163647727518613504	0x47a8f64000000000	47a8f64
12	5163647521360183296	0x47a8f61000000000	47a8f61
13	5163647469820575744	0x47a8f60400000000	47a8f604
14	5163647474115543040	0x47a8f60500000000	47a8f605
15	5163647475189284864	0x47a8f60540000000	47a8f6054
16	5163647475994591232	0x47a8f60570000000	47a8f6057
17	5163647476061700096	0x47a8f60574000000	47a8f60574
18	5163647476078477312	0x47a8f60575000000	47a8f60575
19	5163647476082671616	0x47a8f60575400000	47a8f605754
20	5163647476083720192	0x47a8f60575500000	47a8f605755
21	5163647476083458048	0x47a8f605754c0000	47a8f605754c
22	5163647476083523584	0x47a8f605754d0000	47a8f605754d
23	5163647476083507200	0x47a8f605754cc000	47a8f605754cc
24	5163647476083494912	0x47a8f605754c9000	47a8f605754c9
25	5163647476083493888	0x47a8f605754c8c00	47a8f605754c8c
26	5163647476083493632	0x47a8f605754c8b00	47a8f605754c8b
27	5163647476083493440	0x47a8f605754c8a40	47a8f605754c8a4
28	5163647476083493424	0x47a8f605754c8a30	47a8f605754c8a3
29	5163647476083493420	0x47a8f605754c8a2c	47a8f605754c8a2c
30	5163647476083493423	0x47a8f605754c8a2f	47a8f605754c8a2f

Table 4.8: Binary representation of the ten S2 example cell-IDs from the previous figure

Level	Binary
17	010 0011110101000010100 011100001111111 00000000000000000000000000
15	010 0011110101000010100 011100010010000 00000000000000000000000000
15	010 0011110101000010100 011100010110000 00000000000000000000000000
15	010 0011110101000010100 011111101010000 00000000000000000000000000
14	010 0011110101000010100 100000011000000 00000000000000000000000000
15	010 0011110101000010100 100000100110000 00000000000000000000000000
15	010 0011110101000010100 100011001010000 00000000000000000000000000
14	010 0011110101000010100 100011011000000 00000000000000000000000000
14	010 0011110101000010100 100011101000000 00000000000000000000000000
14	010 0011110101000010100 100011111000000 00000000000000000000000000

Table 4.9: Integer - hex - and token representation of the ten example cells

Level	Binary	Hex	Token
17	5163465904305995776	0x47a850e1fc000000	47a850e1fc
15	5163465905446846464	0x47a850e240000000	47a850e24
15	5163465907594330112	0x47a850e2c0000000	47a850e2c
15	5163466021410963456	0x47a850fd40000000	47a850fd4
14	5163466046107025408	0x47a8510300000000	47a85103
15	5163466053623218176	0x47a85104c0000000	47a85104c
15	5163466141670047744	0x47a8511940000000	47a851194
14	5163466149186240512	0x47a8511b00000000	47a8511b
14	5163466157776175104	0x47a8511d00000000	47a8511d
14	5163466166366109696	0x47a8511f00000000	47a8511f

Table 4.10: Overview and characteristics geo encoding reference systems

Geo encoding system	Characteristics
Long/Lat	Standard for most use cases, two values, 15-20 characters, point location, not suitable for addressing areas
Geohash	12 characters (base32), hierarchical (32/1), Z-order curve, represent areas, semi-regular tiles (rectangular), granularity fine, projection none,
S2	64bit integers, hierarchical (4/1), Hilbert curve, preserving spacial locality, represent areas, semi-regular tiles (quadrilateral projections), granularity fine, spherical projections,

and *sharing phase*. Table 4.11 lists the properties in relation to location privacy for the categories.

Table 4.11: Location privacy properties of four LBS categories by Andersen et al.

	POI	Social Networking	Coll. Sensing	Route Tracing
Receiver	Service	Peers	Service	Peers or service
#Receivers	1	n	n	n
#Users	1	1	n	1
ID importance	High	High	Low	Medium
Spatial Sen.	High	Medium	High	Medium
Temporal Sen.	High	High	High	Medium
Sharing Phase	Online	Online	Online or offline	Offline

4.3.1 Privacy method categories

Andersen and Kjærgaard [140] also name related work for the five privacy categories. The following sections list some protocols and approaches mentioned in the article and add additional ones.

4.3.1.1 Anonymity

In case of anonymity, no identifying information is sent together with the location request. One solution is *k-anonymity* [141], in which dummy clients are created with similar location values. An attacker cannot distinguish between the actual client and the dummy clients. Liu lists in [142] extension *l-diversity* [143] as a location privacy method by reducing the granularity of the data. Other obfuscation methods include *t-closeness* and *differential privacy*; the latter is used, for example, by Apple to obfuscate data on the target client before uploading it to the company's backend servers. For example, differential privacy for geo-indistinguishability is used by Andrés et al. [144].

4.3.1.2 Classical security

Within classical security, encryption and access policies are used to hide information or control its access. For example, the Geospatial eXtensible Access Control Markup Language (GeoXACML) [35] is used to control access to location information. Location-based Access Control (LBAC) adopting GeoXACML policies was previously investigated by Sharhan et al. [34] and Zickau et al. [31]. Figure 4.14 shows which relations of two location objects can be evaluated within an access control policy. GeoXACML PrivLoc is an example of securing geofence information with classical security [145].

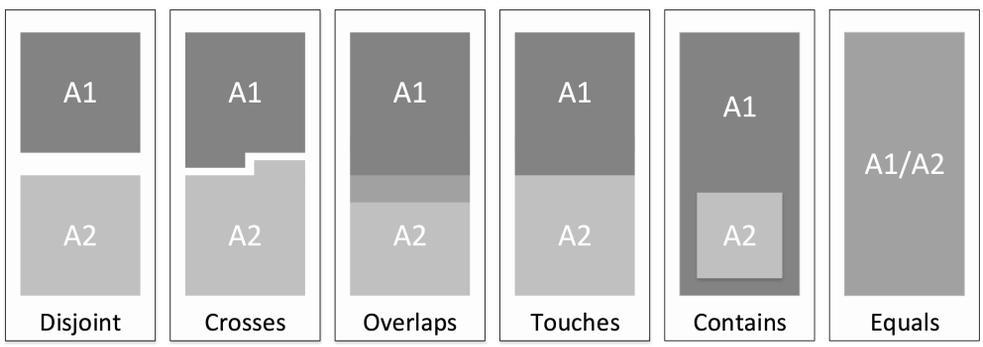


Figure 4.14: Possible location object relations between A1 and A2 that can be evaluated within a GeoXACML access control policy.

4.3.1.3 Spatial and temporal obfuscation

The approaches and methods listed in this section mainly focus on obfuscating spatial or temporal dimensions or increasing the granularity of information to hinder an attacker from knowing the precise location information of target users [146].

The idea behind obfuscation is to present either spatial or temporal fuzzy results to the requester. For example, only a geofence is presented as a location result, which means the target is within the boundaries, but no exact location is revealed. The same holds for temporal obfuscation, where an outdated location is displayed. Using a routing protocol to send the location response through several intermediary clients can help obfuscate the origin client's identity and location [147], usually used in combination with anonymity and spatial obfuscation.

Privacy-aware geofencing in proactive location-based use cases has been researched by Rodriguez Garzon and Deva [78]. They also included other context information for their research, such as time and geofencing sequences [148].

The Casper framework by Mokkel et al. [149] describes a query process for location services without compromising privacy. In Casper, a *location anonymizer* blurs the exact location of a user. The framework also introduces a *privacy-aware query processor* at the database level to deal with the blurred location information and provide the user with the relevant information.

The algorithm PARTS (*privacy-aware routing with transportation subgraphs*) by Roth et al. [150] describes how splitting up routing queries improves privacy queries to a navigation service. The split-up is done on the target client to provide anonymized information with dummy requests to the service.

The PROBE (protecting location privacy against spatial inferences) approach by Damiani et al. [151] takes the geographical context of a target's surroundings into consideration. It introduces *obfuscated spaces* that un-link *geometric* and *semantic* location information. Yigitoglu et al. [152] extended the approach by considering road network constraints. Damiani's work on privacy requirements and research issues in the context of untrusted third party location services and providers [153] motivated this thesis.

4.3.1.4 Protocol

A general model to include privacy policies in Internet applications was the World Wide Web consortium's platform for privacy preferences (P3P) [154]. P3P uses XML policies to be published by websites but is not adopted in practice. One of the early standardized protocols for location privacy is IETF's Geopriv [155], enabling users to specify policies. A context-sensitive privacy-aware framework specifically addressing proactive LBS was published by Deva et al. in 2015 [156].

Contact tracing and exposure notification applications based on the measurement of BLE signals were widely discussed during the 2020 pandemic. Some of these approaches use ephemeral identifiers [111] together with privacy-preserving evaluation protocols on the client. Susan Landau gives a recent overview in the book People Count as of 2021 [60].

In Europe, two protocols for exposure notification apps were competing in 2020, the *Decentralized Privacy-Preserving Proximity Tracing* (DP-3T) protocol [157] and the *Pan-European Privacy-Preserving Proximity Tracing* (PEPP-PT) protocol [158]. Both of these protocols rely on BLE IDs and signals. The fundamental difference between these protocols is where the BLE data is compared and evaluated. PEPP-PT uses a centralized approach (server-side), whereas DP-3T uses a decentralized approach (client-side). As the iOS and Android SDKs only support the DP-3T protocol for privacy reasons, this model was used in most European Corona apps [159] [160].

In general, the article by Michael Herrmann et al. and his thesis provide a comprehensive overview of privacy methods in different categories [80] [161].

4.4 Cryptographic schemes and privacy protocols

This section lists privacy-preserving protocols and schemes addressing location privacy use cases, mainly for non-anonymity and non-obfuscation. The schemes are categorized according to their underlying cryptographic concepts. The concepts described here are *hash functions, Bloom filters, homomorphic encryption, multi-party computation, private*

set intersections, cryptographic accumulators, zero-knowledge proofs, attribute-based encryption, and *private pattern matching*.

4.4.1 Hash functions

Among the approaches used as a privacy-preserving measure, hash functions are fundamental. The following section describes the definitions upon which further functionality is built. If not otherwise stated, the definitions and their formalization are primarily based on Hans Delfs' *Introduction to Cryptography* [90].

4.4.1.1 Cryptographic hash functions

The hash functions used in security applications, such as signature schemes and message authentication codes (MACs), are called *cryptographic hash functions*. Examples are MD5 and SHA256. These cryptographic hash functions can also be used to create *pseudorandomness*, for example, in the form of strings. A cryptographic hash function $h()$ generally takes an arbitrarily long string of bits, the message m, and outputs a fixed-length bit string n. Mathematically it is defined as

$$h() : \{0,1\}^* \rightarrow \{0,1\}^n, m \mapsto h(m). \tag{4.2}$$

The output length n is typically between 128 and 512 bits. Another requirement of a cryptographic hash function is that the value $h(m)$ is easy to compute. However, from a given value $h(m)$, it is computationally infeasible to compute the message m. That is why cryptographic hash functions are also called *one-way functions*.

The most basic security application for a hash function is to *encrypt* a password by storing its hash value instead of the cleartext. In signature schemes, a message is hashed, and then $h(m)$ is signed by the sender, i.e., the owner of a private key. In this context, it is infeasible to compute a second message m' with $m \neq m'$ and $h(m) = h(m')$. This property of hash functions is called *second pre-image resistance*. If such a pair of m, m' exists, a *collision* exists. If it is infeasible to find such a collision for a given m, the hash function is *collision resistant* or *collision free*. There are infinite collisions for mapping an infinite number of messages to a finite number of hash values. The term *resistance* means that they are extremely hard to find. This argument only holds for the pre-quantum computer age. The collision resistance is a stronger property of a hash function than the second pre-image resistance. A second pre-image resistance function is a one-way function, see [90] for details. Finally, a hash function that is collision resistant is a *cryptographic hash function*. Today's widely used algorithms from the SHA-2 family are SHA256 and SHA512. The NIST has already selected the successors from the SHA-3 family [162].

4.4.1.2 Message authentication code

Cryptographic hash functions are also referred to as *message digest functions*. The hash value $h(m)$ of a message m is also called the *digest* or *fingerprint* of m, i.e., it is a very compact and unique representation of a message. This functionality is used to control the integrity of a message. For that, the hash value is stored. At a later stage, it can be compared to a hashed message. This is why hash functions are also called *modification detection codes* (MDC) [90].

For the authenticity and integrity of a message, hash functions are used. For this purpose, the procedures are called *message authentication codes (MACs)*, used for protocols such as SSL/TLS and IPSec [163]. For this purpose, two communication parties, who need a shared secret key k, can produce a valid MAC for a message m. Formally, k is a parameter of a MAC, and therefore a MAC is a constructed family of hash functions, see Equation 4.3.

$$(h_k() : \{0,1\}^* \rightarrow \{0,1\}^n)_{k \in K} \tag{4.3}$$

A MAC is used to authenticate a message from a particular sender and check its integrity. A verifier, also knowing the MAC's secret key, can use it to check the data's integrity and authenticity.

Informally, three algorithms define the usage of a MAC [164].

- $G()$ key generation: creating a random secret key k from a key space n.
- $S()$ signing a message: returning a tag t for a given key k and message string x.
- $V()$ verifying a message: returning *accepted* when the given message x and the key k creates the tag t.

The three algorithms must satisfy the proof (*Pr*), Equation 4.4 [164].

$$Pr[k \leftarrow G(n), V(k, x, S(k, x)) = \text{accepted}] = 1 \tag{4.4}$$

For an unforgeable MAC, it needs to be computationally infeasible to compute a valid tag t for a given message x without the knowledge of the secret key k. MACs are different from digital signatures, as the signing and verification key k is the same and known to more than one party. Digital signatures are based on a *private key* for *signing* and a corresponding *public key* for *verification*. Therefore, digital signatures support *non-repudiation*.

4.4.1.3 Pseudorandom generators

An ideal *perfect hash function* can also be seen as a *random function* [165] that will toss a coin for the hash value $h(m)$ for all bits in the message. In cryptography, this function is also called a *random oracle*. Such a function cannot be implemented because it would require exponentially many ($n \times 2^l$) tosses and storage for all results. It is a design goal and a pragmatic way to construct a hash function that *approximates* a random function, i.e., *pseudorandom* [166]. The objective is that it should be infeasible to distinguish between a *random oracle* and a *pseudorandom function*. Similar design goals also hold for *symmetric encryption keys*. These building blocks are a crucial set of techniques used in this thesis. For a well-designed hash function, $h()$ is a generator of pseudorandom bits. This is also used within the SSL/TLS protocols and the HMAC function [167].

4.4.2 Probabilistic approaches

The following sections overview different *Bloom filters* as data structures used for set membership testing. In Section 4.4.2.4, examples of Bloom filters used for spatial use cases are presented. In Section 5.3.2, it is shown how they can be adapted and expanded

to fit the presented objectives.

4.4.2.1 Bloom filter

In general, a Bloom filter is a *space-efficient probabilistic* data structure which was introduced by Burton Howard Bloom in 1970 [168]. It is used for set membership queries and will be represented by an array of m bits, i.e., the *filter* B, each initially set to 0. $B[i]$ describes the i-th position of the filter. A Bloom filter uses k independent hash functions, $h_j(1 \leq j \leq k)$, where $h_j() : \{0,1\}^* \rightarrow [1,m]$ maps a set element e to one of the m-array positions. When an element e is added to the set S of the specific Bloom filter, the bits in S at the position $B[h_j(e)]$ are set to 1. The set S is defined as $S = \{e_1, ..., e_n\}$.

If a query wants to check if an element x belongs to the pre-defined set S, all k positions in the Bloom filter array need to be 1, $B[h_j(x)], j \in [1,k] = 1$, after applying the independent hash functions. Bloom filters may produce false positives (FP), as described in more detail below, but they do not produce false negatives (FN). The false-positive rate depends on the parameters m, n, and k and, therefore, can be set to a probability level P that suits the use case. The size of the array can be approximated as

$$m = 1/(1-(1-P^{1/k})^{1/kn}). \tag{4.5}$$

The number of hash functions k (a positive integer) and given the values for m and n that minimizes the FP probability is

$$k = \frac{m}{n} \ln 2. \tag{4.6}$$

4.4.2.2 Counting Bloom filter

In 2000, Fan et al. [169] introduced a variant data structure that can delete entries in a Bloom filter without recreating the whole filter. It is known as a *Counting Bloom filter*. Instead of having a single-bit in each bucket, it uses a multi-bit value. The *insert* operation *increments* the value at each relevant bucket, and the *deletion* operation *decrements* the respective values. Typical values for the size of a counter bucket are three or four bits, which increases the overall space needed for storing a Bloom filter. The size of the buckets should be in line with the underlying use case to avoid arithmetic overflows. Different filters with different properties have been proposed over the past years. One of them is described in the subsequent section.

4.4.2.3 Cuckoo hashing

Derived from the family of birds that sometimes push eggs out of another breeding nest, in *cuckoo hashing*, resolving hash collisions leads to the *re-hashing* of values with a second hash function. The values are kicked out by other keys either in the same table into a different cell or by using multiple hash tables and putting collision values, i.e., keys, into those. It was first described by Pagh and Rodler in 2001 [170]. Each data can be put into different locations by using multiple hash functions. The lookup requires inspecting the number of hash functions (and tables) which takes constant time in the worst-case. A greedy algorithm is used for the insertion, kicking out a key if a second key has the same value. The kicked-out data is re-hashed with another algorithm and put

into an alternative cell. This may result in an infinite loop. Different approaches are used when this situation happens, e.g., the whole data might be hashed again using different hash functions or might be put on a stash, an array of a fixed number of keys, in which all values are placed, causing a loop. The latter version reduces the failure rate of the cuckoo hashing. The analysis of cuckoo hashing with a stash extends to practical hash functions [171].

Another advanced version of cuckoo hashing replaces the hash keys with shorter *fingerprints* and applies another hash function to the keys. To allow the movement of the fingerprints without necessarily knowing the key, the two locations of the fingerprints may be computed using a bitwise XOR $\oplus$ operation, either with the fingerprint or with the hash of the fingerprint.

This data structure forms an approximate set membership structure with similar properties to a Bloom filter. It stores the members of a set of keys and can query them. There is also a chance of FP but without one for FN. The advantages over a standard Bloom filter are the constant factor of reduced memory, a better locality of reference, and fast deletion of set data with no storage penalty [172].

The formalized usage of Cuckoo hashing is as follows.

- Choose hashing functions, $h_1(), h_2(), h_3() : \{0,1\}^* \rightarrow [b]$.
- Initialize empty bins $B[1, ..., b]$.
- Hash an item x and check if any of the bins $B[h_1(x)], B[h_2(x)], B[h_3(x)]$ at position b is empty. If so place x in any of the bins and terminate.
- If not, choose a random value $i \in \{1,2,3\}$, take out the item in $B[h_i(x)]$ and replace it with x, for the taken out item recursively find a place in any of the other bins.
- If the process does not terminate after a set number of iterations, the item is placed in a *stash*.

4.4.2.4 Bloom filter as a privacy-preserving scheme

In a 2018 paper, Christoph Bösch [173] describes using a Bloom filter to evaluate geofences in a privacy-preserving way and, as stated, efficiently. The use cases he describes are similar to the ones presented in Section 2.4. In his approach, a trusted third party acts as an intermediary to communicate the evaluated result to the location-based service. In the approach, the data is symmetrically encrypted such that the third party cannot know the position of a target. Ruppel et al. [174] also use Bloom filters to enable a similar scenario. The latter authors use the Geohash encoding method and call their approach *Geocookie*. Palmeri et al. [175] introduced the idea of using spatial Bloom filters for privacy preservation in 2014.

The protocol introduced by Bösch [173] is divided into the following steps.

- The operator O generates two symmetric keys, k_e for en/decryption and k_h for keyed hashing. O also creates the public parameters, *acc* for the GPS accuracy, the encryption algorithm, and the Bloom filter hash functions h_j.
- O generates the Bloom filter array $B_{i/o}$, i.e., the encoded geofences, with the following parameters, k_e, k_h, *acc*, and the geofence in GPS coordinates *pos* with decimal degrees. For all GPS coordinates *pos*, O will
  - encrypt the *pos* with k_e as $E(p) \leftarrow E(k_e, pos)$, with the given decimal digits 1-5 depending on the *acc* value.

  - O will hash $E(p)$ with k_h and set $B[h_j(k_h, E(p))]$ to 1 for all $j \in [1, k]$. The Bloom filter is marked with either *i* for *inside* or *o* for *outside* depending on whether the area is a prohibited geofence.
  - O sends the outputted Bloom filter $B_{i/o}$ to the service provider SP.
- During the evaluation of the position of vehicle V, the protocol implements a monitoring function: $(\text{in/out}) \leftarrow \text{Monitor}(k_e, acc, k_h, B_{i/o})$.
  - Depending on the *acc*, V determines its GPS coordinates regularly and encrypts them with $E(p) \leftarrow E(k_e, pos)$. The result $E(p)$ is sent to the service provider.
- The service provider hashes the input with k_h and queries the Bloom filter $B_{i/o}$. Depending on the outcome of the query, the SP sends a notification to O. If it was previously agreed, the SP could also send the encrypted position $E(p)$ to O, where it can be decrypted to get to know the exact position in case the V is outside of a protected area. The protocol also mentions that this may only be done after V was "spotted" several consecutive times outside the protected area to ensure that no privacy was violated, only because V was at the border of a geofence.

Geocookie Ruppel et al. use Bloom filters for a space-efficient representation of geographic location sets [174]. They named this concept *Geocookie*. As the name suggests, the Geocookie is meant to be a compact data structure stored in a user agent to provide a server with information about *visited* locations. The server can check a location against the Bloom filter structure in the Geocookie and determine if the checked location is part of the filter or not. As Bloom filters allow *false positives* but prevent *false negatives*, the service gets to know that the agent has not been at the location or that the agent was at the location with a high probability. The Geocookie uses the Geohash encoding system [120] to store location information. As the set in the filter is aggregated and held by the client, the concept is also privacy-preserving. Ruppel et al. also mention that the approach can check distances between user agents. In this case, a function checks if one of the eight neighboring cells of a location is part of the Geocookie. The publication also mentions different types of Geocookies, *browser-based*, *local*, *hierarchical*, *public*, and *concealed* ones. The grid cell geo encoding and its set membership testing for privacy-preserving geofencing are also applied to Bloom filter schemes, as presented in Section 5.3.2, and are influenced by Geocookies.

4.4.3 Homomorphic encryption

With a *homomorphic encryption* (HE) scheme, it is possible to make calculations on encrypted data. The results as plaintext can be examined after decryption. Only basic mathematical functions can be used in HE. Just to illustrate such a scheme, we use two simplified $enc(x)$ and $dec(x)$ algorithms.

- $enc(x) = 2x$
- $dec(x) = x/2$
- The plaintexts are: $a = 3$ and $b = 5$
- Encryption of a and b: $enc(a) = 6$ and $enc(b) = 10$
- Addition computation on the ciphertext: $6 + 10 = 16$
- Decryption of the resulting ciphertext $dec(16) = 8$
- The same result as the addition of the two plaintexts a and b: $3 + 5 = 8$

A *fully homomorphic encryption* (FHE) scheme would allow arbitrary computation on

ciphertexts. Such schemes would help offload heavy computational tasks to a cloud computing service without revealing any potentially sensitive data to the external service. It is also called outsourced computation to enable new services by removing privacy barriers inhibiting data sharing.

In Appendix A.2, several homomorphic encryption schemes and their respective homomorphic properties used in this thesis are described in detail.

NEXUS The non-exposure user location privacy System (NEXUS) by Guldner et al. [176] uses the partially homomorphic *Paillier* encryption scheme, as described in Appendix A.2.2 for its implementation. Within the NEXUS system and the limitations of HE, the geofence is a rectangle with four long/lat values (A,B,C,D), see Figure 4.15. This approach will also be extended to support arbitrary shapes as it is also a privacy-preserving geofencing mechanism, see Section 5.3.3. Figure 4.16 shows the workflow of the protocol.

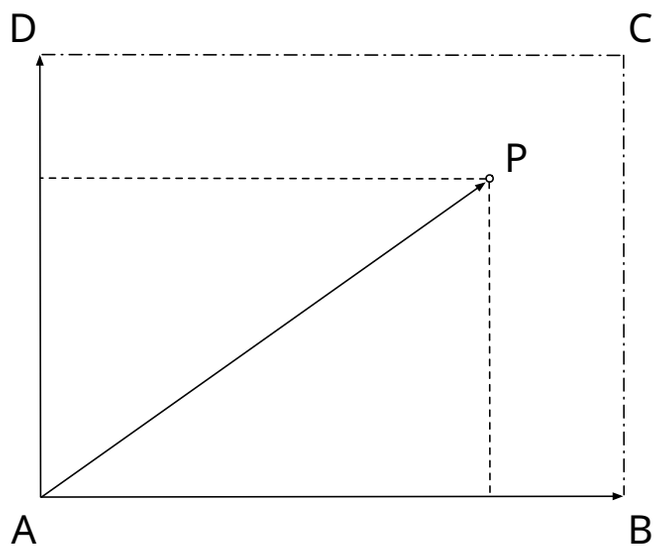


Figure 4.15: NEXUS homomorphic encryption vector approach.

The following steps are made to achieve the privacy evaluation.

1. The *Key Authority* (KA) generates a public key (pk) and a secret key (sk) pair. The KA sends the public key to the target client, i.e., the *mobile node* and the geofence service, i.e., the TTP. The geofence service holds the geofence that needs to be evaluated. The geofences are encrypted with the public key of the KA.
2. The target client calculates its long/lat position and encrypts it with the public key of the KA. The target client sends this encrypted value to the geofence service (TTP).
3. The geofence service calculates the result using the HE scheme with the encrypted geofence and the encrypted position. For that, each corner of the rectangle is evaluated against the client's position in HE.

A value is computed by subtracting these geofence coordinates from location coordinates for each cardinal direction (north, south, west, east). These values are still encrypted and cannot be revealed without the private key. This is the most computation-intensive part of the NEXUS project. Afterward, these four computed encrypted values are transmitted to the evaluator. The evaluator can decrypt these values by using the private key. By using revealed subtraction results, it is computed whether the client location is inside the geofence or not.

4. The results are sent to the KA.
5. The KA is the only party that can decrypt this HE result as it holds the private key. The KA only learns if the location is inside or outside the given geofence.
6. The KA sends the computed result to the *subscription nodes*, i.e., if the client is within the geofence or not.

In this approach, it is the geofence service that does not know the position of the mobile device. It has to be emphasized that the TTP plays an indispensable role in this approach. Otherwise, the geofence service would also hold the decryption key and decrypt the position upon receiving it. This also means that the TTP and the KA could gain access to the private position of the target if they worked collaboratively.

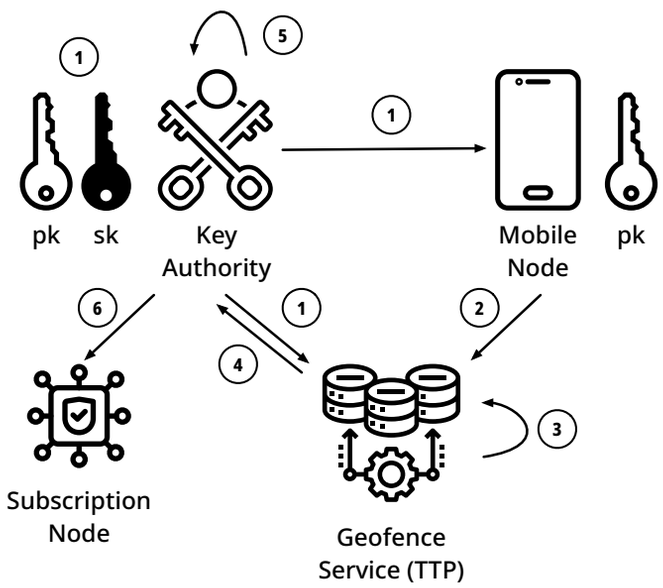


Figure 4.16: NEXUS architecture overview.[10]

Again, it has to be emphasized that the approaches presented in this thesis are mainly designed to be independent of (trusted) third parties.

4.4.4 Multi-party computation

Secure *multi-party computation* (MPC), also known as *secure computation* or *privacy-preserving computation*, is the field of applied cryptography protocols that create methods for the mutually distrusting participants to compute a joined function but with keeping each input parameter private. The types of computation in MPC determine the output of many inputs. It contrasts with traditional methods of securing messages between sender and receivers. The initial use cases were playing games over greater distances without relying on a trusted third party. David Evans et al. believe that MPC usage in privacy-preserving applications is forthcoming, as stated in their publication on the topic in 2018 [177]. They also proclaim that MPC is currently used for enabling data sharing.

In the presented use cases, the target client and the service want to compute the outcome of the geofencing function without actually revealing the input parameters. Depending on the use case, the geofence is either known to all parties or hidden from specific roles.

[10]Image source: The figure was adapted from the original publication by Guldner et al. (2018).

In general, there are a given number of participants, e.g., $p_x = [p_1, p_2, ..., p_n]$, each with their private data inputs, $d_x = [d_1, d_2, ..., d_n]$. Each p_x wants to compute the output of a public function $f()$ over their combined inputs, i.e., $f(d_1, d_2, ..., d_n)$.

Suppose there are three users, Andy, Becky, and Charlie. They want to compute the person with the highest pocket money without revealing how much each person gets. In this case, the public function is $f() = max(d_1, d_2, d_3)$. Obviously, this could be done using a *trusted third party*. The goal of MPC is to design a protocol in which the three actors exchange messages and do not learn anything more than the outcome of the public function without the need for a TTP.

The most basic properties an MPC ensures are the following.

- **Input privacy**: The only information about the private input data can be inferred from the outcome of the function $f()$.
- **Correctness**: Any party or parties deviating from the protocol or working jointly together can force an honest party to output a false result. This comes in two flavors. Either the honest parties are guaranteed to compute the correct result (*robustness*), or the protocol aborts if it finds an error (*with abortion*).

Besides the presented use cases regarding computational geometry, there is a wide range of practical usages for MPC protocols, such as electronic auctions, electronic voting, and privacy-preserving data mining.

In the presented use case of privacy-preserving geofencing, there are two parties involved, the target client and the service. *Two-party computation* (2PC) protocols may differ from those of MPC. However, MPC is of value in the use case of privacy-preserving distance measuring between multiple targets or in location-based social networks and community services.

4.4.5 Private set intersection

In a general MPC protocol, the elements of the two sets themselves may be revealed to one or two parties. In our use cases, only the knowledge of the number of elements, i.e., the *cardinality*, or even if there is one element in common, i.e., the *existentiality*, is necessary for the receiving party to know. The formal definitions of the two variants of a *private set intersection* protocol are described in the following subsections [132].

If the set elements are equal in number and value, the parties would learn all the intersection protocol values. This might not be the desired outcome. Hence, learning a function about the intersection might be advantageous, such as the described PSI cardinality function below.

4.4.5.1 PSI cardinality

Private set intersection cardinality (PSI-CA) is a protocol between a server set with w items $S = \{s_1, ..., s_w\}$ and a client set with v items $C = \{c_1, ..., c_v\}$. At the end of the execution, the client learns exactly $(|S \cap C|, |S|)$, and the server learns exactly $|C|$. PSI-CA securely implements $f_{\text{PSI-CA}} : (S, C) \rightarrow (|C|, (|S \cap C|, |S|))$. PSI-CA is based on Cristofaro et al. [178], which is also the basis for Faber's variant [132].

4.4.5.2 PSI existentiality

An even more restrictive variant of the standard PSI protocol is that it only reveals to the client and server if there is a common element in both sets. This can be a boolean value indicating if the two sets intersect or not. From a privacy perspective, this is the desired functionality in the context of the thesis.

Faber [132] and Carpent et al. [179] describe a solution for *private set intersection existentiality* (PSI-X), which is based on *fully homomorphic encryption* (FHE). PSI-X only tells the parties or one party if two sets, $A = \{a_1, ..., a_n\}$ and $B = \{b_1, ..., b_m\}$ with different numbers of elements, have at least one element in common. It would not reveal any information about what element(s) or the cardinality. Hiding the cardinality would improve the privacy-preserving functionality in our grid cell evaluation and satisfy unlinkability. The FHE approach would have quadratic complexity and is described as follows. FHE is used to aggregate values of encrypted data.

$$y_{ij} = E(r_{ij} * a_i - b_j) \text{ for each } i \in [1, n], j \in [1, m] \text{ and a unique random } r_{ij} \tag{4.7}$$

$$E(\Pi_{i=1,j=1}^{n,m}(y_{ij})) \tag{4.8}$$

The product in Equation 4.8 would decrypt to zero if $A \cap B \neq 0$ or to a random value otherwise. The heart of this approach can be described as if the two sets have at least one member in common. These two members would generate a value of $y = 0$ and build the product of all y_{ij} would be zero. One problem is that by constructing the y_{ij} and encrypting them, this party would know the members. The party constructing the encryption of the factors y_{ij} would know how many of them would be zero and therefore know the cardinality. The values would need to be encrypted before building the single products y_{ij}, leading to unlinkability issues *again*.

As FHE is impractical as of this writing, Faber [132] and Carpent et al. [179] presented an alternative construction, which is described and adapted in Section 5.3.4.3.

4.4.6 Cryptographic accumulators

Josh Benaloh and Michael de Mare originally introduced the notion of one-way accumulators in 1993 [180]. The overall characteristics of an accumulator are

- One-way function that satisfies a quasi-commutative property,
- No trusted authority is needed,
- They are used for time-stamping and membership testing.

The summed-up cryptographic characteristics of accumulators are

- Correctness,
- Collision freeness $\Rightarrow$ binding,
- Undeniability,
- Indistinguishability $\Rightarrow$ hiding,
- Perfect completeness $\equiv$ correctness,
- Soundness $\equiv$ undeniability, and
- Zero-knowledgeness.

Section 4.4.7 below explains the zero-knowledgeness characteristics.

4.4.6.1 One-way accumulators

A binary operation is *commutative* if one can change the operands' order without changing the operation's result. The most famous examples are multiplication and addition, e.g., $6 \cdot 7 = 7 \cdot 6$ or $11 + 12 = 12 + 11$. Of course, division and subtraction are not commutative operations, also referred to as *non-commutative* operations. The corresponding property of binary relations is called *symmetric*. Examples of commutative binary operations are the addition of vectors and the intersection and union of sets. Matrix multiplication of square matrices is almost always non-commutative.

The following definitions are adapted from Nelly Fazio et al.'s cryptographic accumulator article from 2002 [181].

A function $f() : X \times Y \rightarrow X$ is said to be *quasi-commutative* if

$$(\forall x \in X)(\forall y_1, y_2 \in Y)[f(f(x, y_1), y_2) = f(f(x, y_2), y_1)]. \tag{4.9}$$

A family of *one-way hash functions* is an infinite sequence of families of functions $\{\mathcal{H}_\lambda\}_{\lambda \in \mathbb{N}}$ where $\mathcal{H} = h_k() : X_k \times Y_k \rightarrow Z_k$, with the following properties

1. For any integer λ and any $h_k \in \mathcal{H}_\lambda, h_k(\cdot, \cdot)$ is computable in time polynomial in λ.
2. For any probabilistic, polynomial-time algorithm $\mathcal{A}$

$$PR[h_k \overset{R}{\leftarrow} \mathcal{H}_\lambda; x \overset{R}{\leftarrow} X_k; y, y' \overset{R}{\leftarrow} Y_k; x' \leftarrow \mathcal{A}(1^\lambda, x, y, y') : h_k(x, y) = h_k(x', y') < negl(\lambda)] \tag{4.10}$$

where the probability is taken over the random choice of $h_k(), x, y, y'$ and the random coins of $\mathcal{A}$.

Definition [180]: A family of *one-way accumulators* is a family of *one-way hash functions*, each of which is *quasi-commutative*.

There are different accumulator schemes, each iteration adding more functionality. This thesis uses the following three accumulator schemes in the context of the privacy-preserving geofencing approach.

4.4.6.2 Elementary accumulators

An accumulator scheme, also referred to as *elementary accumulators*, defined in [181] and originated in [182], is a 4-tuple of polynomial time algorithms (*Gen*(), *Eval*(), *Wit*(), *Ver*()).

- *Gen*() is a key generation algorithm to set up the parameters. The security parameter 1^λ and a threshold N. N is the upper bound of the total number of values to be accumulated. *Gen*() returns an accumulator key k from a keyspace $\mathcal{K}_{\lambda,N}$.
- *Eval*() is an evaluation algorithm used to accumulate a set of $L = \{y_1, ..., y_{N'}\}$ of $N' \leq N$ elements from a domain Y_k. k is the accumulator key. *Eval*()'s input is

$(k, y_1, ..., y_{N'})$, it's return value $z \in Z_k$, i.e., the accumulator, together with some auxiliary information *aux*. *aux* will be used in the witness algorithm *Wit*(). Every *Eval*() with the same input must produce the same *z*, the *aux* value can differ.

- *Wit*(), a witness algorithm is a probabilistic algorithm with input parameters k, a value y_i, and the auxiliary information *aux*. It returns a witness $w_i \in \mathcal{W}_k$ which proves that y_i was accumulated within z if this is the case or the special symbol $\perp$ if $y_i \notin \{y_1, ..., y * (N')\}$.
- *Ver*() is a deterministic verification algorithm which returns *yes* or *no* on input (k, y_i, w_i, z) whether the witness w_i embodies a valid proof that y_i has been accumulated within z or not.
- Efficiently computable $\forall x \in acc_x$.
- Intractable to compute $\forall x \notin acc_x$.

The last two characteristics are mutually opposing for *universal accumulators*, see the respective section below.

4.4.6.3 Dynamic accumulators

A dynamic accumulator scheme, defined in [183], extends the 4-tuple scheme to a 7-tuple of polynomial time algorithms (*Gen*(), *Eval*(), *Wit*(), *Ver*(), *Add*(), *Del*(), *Upd*()).

- *Gen*(), *Eval*(), *Wit*(), and *Ver*(), these are defined as in the elementary accumulator in the previous section.
- *Add*(), an element addition algorithm is a usually deterministic algorithm, which takes the values, the accumulator key k, the accumulator $z \in Z_k$, and a new element $y' \in Y_k$. It returns a new accumulator $z' \in L \cup y'$ together with a witness $w' \in \mathcal{W}_k$ and an updated aux_{Add}, that will be used in the *Upd*() algorithm.
- *Del*(), an element deletion algorithm. A deterministic algorithm usually uses the key k, a value $z \in Z_k$, and an element $y' \in L$. It returns a new accumulator z' corresponding to $L \backslash y'$ together with the update information aux_{Del}, that will be used in the algorithm below.
- *Upd*(), is a witness update algorithm, mostly deterministic, that updates the witness $w \in \mathcal{W}_k$ for an element $y \in Y_k$, which was previously accumulated with an accumulator $z \in Z_k$ after *Add*() or *Del*() was executed for an element $y' \in Y_k \backslash \{y\}$ in (or from) z. *Upd*() takes as input (k, y, w, b, aux_{op}), *op* being *Add*() or *Del*(). It returns an updated witness w' that proves the presence of y within the updated accumulator z'.

The definition above is in line with the presentation in [181]. For example, it does not rely on a trapdoor value t_k as it was the case with the original definition in [183]. The value t_k would depend on a trusted central authority, i.e., an *accumulator manager*, which has to be trusted with the knowledge of the value t_k. The schemes needing a trapdoor value are called *trapdoor dynamic accumulator schemes*. This thesis aims to make trusted third parties obsolete so dynamic accumulator schemes are used without trapdoor information.

4.4.6.4 Universal accumulators

Derler et al. [184] add the following properties to *elementary* and *dynamic accumulators* to realize *universal accumulators*. For elementary accumulators schemes, the *Wit*() and the *Ver*() algorithms alter by adding a *type*, indicating whether the given witness is a

membership (*type* $= 0$) or a *non-membership* (*type* $= 1$) witness. This alteration is also applied to the *Upd*() algorithm for dynamic accumulators.

The same authors also define two types of *indistinguishability* for accumulators, namely one that depends on inherently randomized constructions and one that relies on a transformation. The transformation is defined as follows.

- On input of a set $\mathcal{X}$, the *Eval*() algorithm samples an element $x_r \notin \mathcal{X}$ at random from the accumulator domain. After that it computes and returns $(aux_{\mathcal{X}'}, aux')$ for $\mathcal{X}' \leftarrow \mathcal{X} \cup x_r$ and $aux' \leftarrow (aux, x_r)$.
- Demonstrate non-membership
- Non-membership witness $\underline{wit}_x$
- Efficiently computable $\forall x \notin acc_x$
- Intractable to compute $\forall x \in acc_x$

Sections 5.3.5 and 7.2 show how the different types of CA are used to address the thesis' objectives.

4.4.7 Zero-knowledge proofs

An interactive proof system should not transfer any knowledge from the prover *P* to the verifier *V*. Such a proof system is called *zero-knowledge* (ZK). The ZK property is a property of the prover *P*. It secures the prover's input (*P's knowledge*) against honest and even dishonest verifiers [90].

The following order of notions of the general concept of *proofs of knowledge* exemplifies the history of proof systems

1. Proof of knowledge (PK),
2. Zero-knowledge proof (ZKP),
3. ZKP of knowledge (ZKPOK),
4. Interactive ZKPOK (IZKPOK),
5. Non-interactive ZKPOK (NIZKPK).

The goal that needs to be achieved to address the use cases described in Chapter 2 is to construct a *non-interactive zero-knowledge proof of knowledge*. In our case, the *knowledge* of the target is its position in relation to one or more geofences.

4.4.7.1 From proof of knowledge to zero-knowledge proofs

The Camenisch and Stadler [185] notation for proof of knowledge (PK) is

$$\mathrm{PK}\{(\delta, \gamma) : y = g^{\delta} h^{\gamma} \wedge (u \leq \delta \leq v)\}. \tag{4.11}$$

This y is the commitment to the secret δ, which is contained in the interval $[u, v)$. The Greek letters are used to denote values that are only known to the prover, i.e., the target client. In this example, δ is the private data, and γ is a random value that hides δ.

A zero-knowledge proof scheme has the following properties.

Completeness If the knowledge statement is true, an honest verifier will be convinced of this fact by an honest prover.

$$\begin{aligned} &\forall k \in \mathit{Setup}(1^k), \forall (y,w) \in R_\sigma \\ &\mathit{Pr}[\pi \leftarrow \mathit{Prove}(\sigma,y,w) : \mathit{Verify}(\sigma,y,\pi) = \text{accept}] = 1 \end{aligned} \tag{4.12}$$

Soundness If the knowledge statement is false, no cheating prover can convince the honest verifier that it is true, usually with high probability.

For every malicious prover, $\tilde{P}$, there exists a negligible function, v, such that

$$\begin{aligned} &\Pr[\sigma \leftarrow \mathit{Setup}(1^k), (y,\pi) \leftarrow \tilde{P}(\sigma) : \\ &y \in L \wedge \mathit{Verify}(\sigma,y,\pi) = \text{accept}] = v(k). \end{aligned} \tag{4.13}$$

By increasing k, the soundness error can be made arbitrarily small. There is *perfect soundness* if the soundness error is 0 for all k. If the witness w does not satisfy x, then the probability

$$\text{Prob}[\mathit{Verify}(\mathit{Prove}(x,w))] = 1 \tag{4.14}$$

is sufficiently low.

Zero-knowledgeness If the *statement is true*, the verifier learns *nothing more* than the fact that the statement is true, i.e., the verifier *does not know the secret* or *anything derived from it*.

Only knowing the statement (not the secret) is sufficient to imagine a scenario showing that the prover knows the secret. This is formalized by showing that every verifier has some *simulator* that, given only the statement to be proved, can produce a *view* that *simulates* the interaction between the honest prover and the verifier.

A *view* is the interaction between the prover and the verifier. Having a polynomial-time *simulator*, which has access to the verifier's input to generate a *simulated view*, a ZKP scheme is *perfect zero-knowledge* if the simulated view has the same distribution as the original view. A ZKP scheme is *statistical zero-knowledge* if the distribution is *statistically close* and *computational zero-knowledge* is if there is no polynomial-time distinguisher for those distributions [186]. We call it *proof of knowledge* if there is an *extractor*, which has *rewindable* black-box access to the prover and can compute the witness w with non-negligible probability.

4.4.7.2 Non-interactive zero-knowledge proofs

A non-interactive zero-knowledge proof (NIZKP) scheme is defined by three algorithms: *Setup*(), *Prove*(), and *Verify*() as follows.

- *Setup*(): A setup algorithm generates parameters.
- *Prove*(): The algorithm receives an instance x of some NP-language L and the witness w and outputs the *proof*.
- *Verify*(): The algorithm receives the *proof* as input and outputs the value 1 if the verifier accepts the proof and 0 otherwise.

In this model, the prover and the verifier are in possession of a *common reference string* (CRS) taken from a distribution D via a trusted setup

$$\sigma \leftarrow Setup(1^k). \tag{4.15}$$

To prove a statement $y \in L$ with witness w, the prover runs

$$\pi \leftarrow Prove(\sigma, y, w) \tag{4.16}$$

and sends the proof, π, to the verifier.

If

$$Verify(\sigma, y, \pi) = \text{accept} \tag{4.17}$$

the verifier accepts the proof or otherwise rejects it.

The witness relation can be generalized to

$$(y, w) \in R_\sigma. \tag{4.18}$$

Sections 5.3.6 and 7.2 show how ZKPs are used to address the thesis' objectives.

4.4.8 Attribute-based encryption

Besides, new cryptographic schemes and protocols, such as *homomorphic encryption*, *attribute-based encryption (ABE)*, a subset of *functional encryption* schemes [187], has been proposed. These ABE schemes include a Boolean access policy within the ciphertext, known as *Ciphertext-Policy-ABE* (CP-ABE), or within the secret key, *Key-Policy-ABE* (KP-ABE), respectively. As functional encryption schemes are computationally expensive and therefore not always ready for a production environment, subset ABE implementations are more suitable for today's use cases, e.g., within IoT, see Touati and Challal [188] and Thatmann et al. [189] or in secure cloud computing scenarios, see Zickau et al. [190]. The thesis will provide a comprehensive look at applying ABE within privacy-preserving geofencing. These contexts also demand the introduction of dynamic attributes, for example, position-dependent or time-based information.

ABE is a family of asymmetric encryption schemes. The first ABE scheme was derived by Sahai and Waters and published by Goyal et al. in 2006 [191]. One of the early implementations, called *cpabe*, was developed by Bethencourt in 2007 [192].

A short introduction to the main aspects and definitions based on this CP-ABE implementation is given here. In ABE schemes, there are three different types of keys. A *key authority* generates a *secret master key* and an associated *public key*. The key authority needs to be a trusted entity. *Private keys*, mostly known as *secret keys*, are the third type of key and can decrypt data encrypted by a corresponding public key. The secret master key is used to generate private keys, i.e., secret keys.

An access control policy comes into play when an entity encrypts a *message*. In this process, a *Boolean policy*, consisting of *attribute strings*, *numbers*, and *operators*, is

included within the *ciphertext*. A private key is needed to successfully decrypt the message, including the attributes satisfying the policy within the ciphertext. The Boolean policy can be a fine-grained access control policy, incorporating *attribute sets*. Such a policy could look like: *((student and enrollment < 2018) or professor)*. This access policy describes that the encrypted message, i.e., a file, can only be decrypted by entities with a private key with the attributes *student* and *enrollment<=2017* or *professor*. Only with these attributes access to the plaintext is possible.

One advantage of ABE is decoupling the processes of *private key creation* and *decryption*. Private keys fulfilling the policy can always decrypt data with a matching policy. The private key entity does not need to communicate with the key authority again to decrypt new encrypted messages. This is a feature in use cases in which revocation of keys or attributes is required, and consequently, a re-encryption of data or re-distribution of keys is necessary. This behavior is addressed by schemes using a proxy [193] or policy delegation [194], which allows updating the access policy to a more restrictive one, only using the public key.

ABE includes the necessary and essential feature of preventing *collusion attacks*. For example, the message is encrypted with the policy *(Y and X)*, the user *A* has a private key with the attribute *Y* while user *B* has a private key with the attribute *X*. Neither can decrypt the file, as they both have one needed attribute nor can they combine their keys to decrypt the file, as ABE prevents this collusion resistance, as described in [192].

4.4.8.1 Functions

As mentioned before, the four basic functions in CP-ABE are seen below. The *delegate* function is an additional *nonbasic* one.

- *Setup*(): Generates a *public key* and an associated *secret master key*.
- *Keygen*(): Creates a *private key* with *attributes*. To do this, a secret master key is needed. This private key is associated with the secret master key and its public key.
- *Encrypt*(): Encrypts a file with a *public key* and an annotated *policy*.
- *Decrypt*(): Needs a *ciphertext* and a *private key*. Decrypts the ciphertext only if the attributes in the private key satisfy the policy and the private key is associated with the public key that has been used initially for encryption. Policies and attributes switch places in KP-ABE.
- *Delegate*(): Only a *private key* is needed to generate a new one with a *subset* of the *initial attributes*.

The functions for KP-ABE are defined accordingly.

4.4.8.2 Attributes, relations, and operators

There are two types of atomic formulas: *normal* and *numerical* attributes. A normal attribute is a *string of letters and digits* that begins with a letter, e.g., *student*, *professor*, or *workgroup303*. Numerical attributes have a *name* and an *associated value*, e.g., *hiringdate=2014*, *yearofbirth=1990*. In Bethencourt's cpabe, integers in the range from 0 to $2^{64} - 1$ are supported [192].

In access policies, these numerical attributes can be compared to integers using the following relations: =, <, >, <=, >=. Examples are *hiringDate<=2020*, *yearOfBirth=1999*,

accessLevel>=8. It is not possible to compare two different numerical attributes of a user, i.e., the following predicate is not feasible: *studentCount>examinationCount*. The second argument of a numerical relation comparison always needs to be known at the time of encryption.

For the access policy itself, there are three possible operators: *or*, *and*, and *of*.

- *or*: $formula_1$ *or* $formula_2$ evaluate to true when *either* of the formulas is true.
- *and*: $formula_1$ *and* $formula_2$ evaluate to true only when *both* formulas are true.
- *of*: *k of* ($formula_1$, $formula_2$, ..., $formula_n$) evaluates to true only when *at least k* formulas are evaluated to true with $k \leq n$.

These operators can be combined to form more fine-grained formulas. A file might be encrypted with the following access policy: *1 of (((student) and (enrollment < 2014) and (termsStudied > 3)), ((student) and (termsStudied > 5)), (professor))*.

ABE is intended to be deployed in a *key encapsulation scheme*, such as asymmetric encryption algorithms, i.e., a *hybrid encryption* scheme [90]. This is the standard way to construct systems using asymmetric encryption, where the asymmetric part is computationally expensive. This is also necessary for ABE because of the complexity and the time needed to generate keys and encrypt or decrypt ciphertexts.

4.4.8.3 Attribute-based encryption and dynamic attributes

Several publications have discussed adding *dynamic attributes* to ABE [195] [196]. However, each of these findings has some drawbacks. These findings were researched in a supervised master thesis on location-based access control. They were previously published in Denisow et al. [197].

In Weber's approach from 2009 [195], the *dynamic attributes* are inserted into a specific branch in the *policy tree*. Only the *static attributes* are saved in a static subtree, whereas the *dynamic attributes* are saved in a dynamic subtree. *Both* subtrees need to evaluate as *true* because they are combined with an *AND*. These *dynamic attributes* are actually part of the policy, but only this single construction is made to handle *dynamic attributes*. This reduces the expressiveness of ABE by requiring that the *dynamic attributes* need to evaluate positively, or the file cannot be decrypted. An example of a formula that would not be possible to implement using this approach *(manager or (employee and (time > 8 and time < 16)))* where *time* is a *dynamic attribute*.

In 2012, Weber [198] proposed a way to combine *location-based encryption* (LBE) [199] and ABE. This solution does not integrate the dynamic information into the *attributes* or the *policies*. It acts as an additional layer on top of the normal encryption and decryption. ABE is used in a hybrid encryption scheme and encrypts a relatively small amount of data. This encrypted data is used as a *symmetric key* for the encryption or decryption using symmetric algorithms, e.g., the *advanced encryption standard AES* [200]. In the proposed addition of dynamic locations to ABE by Weber, this *dynamic key* is XORed with the result of the LBE, resulting in a final *key* that is then used for encryption and decryption.

A problem with this solution is that *all* encrypted files using this method require users to be at the specified location. It is not possible to make this dependent on a *part of the formula*. For example, the policy *manager or (employee and location:office)* describes that

a file can only be decrypted if the person is a manager or if the person is an employee that is currently in her office. This policy cannot be represented with Weber's approach.

4.4.9 Secure string matching

The primitives of *secure pattern matching* (SPM) and *secure substring matching* (SSM) are two-party protocols in which one party holds a text t and the other a pattern p. The protocol reveals information to the pattern holder if the pattern matches passages in the text. The party defining the pattern can use wildcards to match any character. The following notation is adapted from Faber [132].

- Σ is the alphabet of allowed symbols.
- The text t of length n is constructed from the alphabet, i.e., $t \in \Sigma^n$.
- The pattern p of length m is also constructed from the alphabet, i.e., $p \in \Sigma^m$.
- $t[j:l]$ indicates a substring t starting at position j with length l.
- $\overline{t_i}$ is representing a substring at position i.
- $x = (x_1, ..., x_m)$ is equal to $y = (y_1, ..., y_m)$ if and only if for all $i \leq m$, $x_i = y_i$.
- $w = *$ is the wildcard character which evaluates to true for any $w = x$. w is not part of the alphabet, i.e., $w \notin \Sigma$.

In the literature, the primitives are called *secure*. This thesis focuses on the privacy aspects of these security protocols. For that reason, the term *private pattern matching* is used as an umbrella term for the following definitions as described in [132].

4.4.9.1 Secure pattern matching

Secure pattern matching (SPM) is a protocol between a client and a server. The client's input is pattern $p = (p_1, ..., p_m)$. The server's input is the text $t = (t_1, ..., t_n)$. t_i are letters from the alphabet. Any p_i can be the wildcard character w or a letter from Σ. The SPM protocol securely implements

$$f_{\text{SPM}}(p,t) \rightarrow (m,(b,n)), \text{where } b = \begin{cases} 1 & \text{iff } \exists ss.t.t[s:m] = p \\ 0 & \text{otherwise.} \end{cases} \tag{4.19}$$

The client learns n and b if $t[s:m] = p$ for any $s < n$. The server learns m.

4.4.9.2 Secure substring matching

In the variation *secure substring matching* (SSM), the pattern p must match exactly one location. Analog to the previous protocol, the client's input is pattern $p = (p_1, ..., p_m, s)$. The server's input is the text $t = (t_1, ..., t_n)$. t_i are letters from Σ. Any p_i can be the wildcard w or a letter from the alphabet. The SSM protocol securely implements

$$f_{\text{SSM}}((p,s),t) \rightarrow (m,(b,n)), \text{where } b = \begin{cases} 1 & \text{iff } \exists t[s:m] = p \\ 0 & \text{otherwise.} \end{cases} \tag{4.20}$$

The client learns n and b if $t[s:m] = p$ for any $s < n$. The server learns m. These secure schemes are used in Section 5.3.8 as *privacy-preserving* schemes.

4.5 Chapter summary

The chapter gave an overview of the related work and state-of-the-art. It started with presenting different point-in-polygon algorithms and geo encoding systems. This was chosen to be presented as they differ from the approaches introduced in the following chapter. The chapter continued with the presentation of various privacy-preserving methods, schemes, and protocols and their categorization.

I'm walking here! I'm walking here!

Midnight Cowboy (1969)

5 Privacy-preserving Geofencing

In this concept and design chapter, the main contributions are the selection, definition, and modification of geofence encodings and the adaptation of cryptographic primitives and schemes for *privacy-preserving point-in-polygon* evaluation (P3IP) to meet the use cases and objectives presented in the motivation chapter.

The geofence and position encodings are based on the *S2* and *Geohash* representations. The cryptographic schemes and primitives for P3IP are based on *cryptographic hashing*, *probabilistic methods*, *homomorphic encryption*, *multi-party computation*, *private set intersection*, *zero-knowledge proofs*, *cryptographic accumulators*, *attribute-based encryption*, *private pattern matching*, and *policy-based algorithms* and *protocols*. The term *privacy-preserving geofencing* or PPG is used throughout the thesis as a broader term for P3IP.

5.1 General concepts

This section gives a short overview of the different *concepts*, their *layers* and *instances*, and the chosen privacy *approaches*. Additionally, the notions of *privacy* and *security* are presented, emphasizing their distinctions in the context of the thesis. This is followed by arguments why the *intuitive* approach of letting the mobile device alone evaluate its position against geofences is not addressing all objectives. The section concludes by presenting the generic steps of a *cryptographic* approach.

5.1.1 Concept layers, instances, and approaches

Figure 5.1 shows the concept layers. These five layers are *hardware*, *positioning*, *privacy and cryptography*, *application*, and *interface*. It starts at the bottom with the hardware layer. This layer describes the hardware to obtain a position, such as a GNSS or Cell-ID module. The layer above is the positioning and point-in-polygon layer. It consists of the choice of a base geodetic standard, such as WGS84. The layer also includes the geo encoding of points and polygons, such as S2 or Geohash, and the PiP evaluation method.

The privacy and cryptography layer includes the cryptographic scheme that enables privacy-preserving data evaluation. The privacy features are summarized in the second to last layer depending on the use cases and application objectives. The motivation chapter showed that different application domains result in different privacy features. On top of it, the user interface layer is displayed.

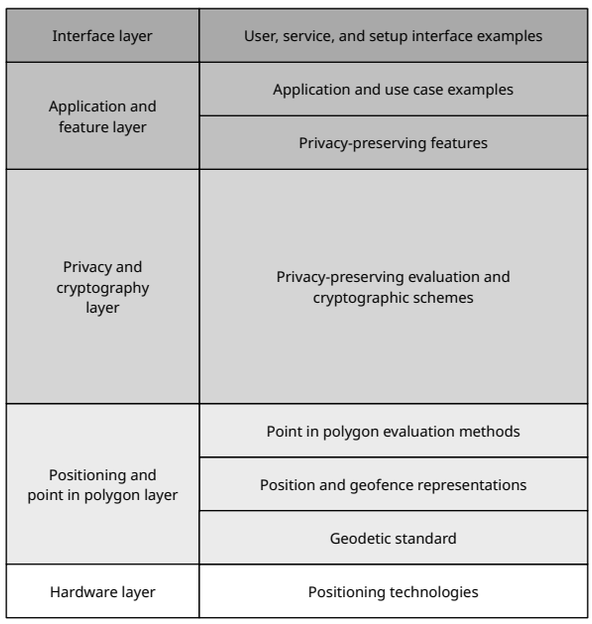


Figure 5.1: Concept layers and exemplified descriptions

Figure 5.2 shows the approaches and instances presented starting with Section 5.2 for positioning and polygon evaluation. The thesis uses S2 and Geohashes as its main positioning representation because it enables a set membership query based on a child-position-parents evaluation which suits the cryptographic privacy methods. This section also introduces the inverse geofence approach, which schemes can utilize. The cryptographic method approaches start with the fundamental hashing schemes in Section 5.3.1. Probabilistic methods, which include Bloom filters, are presented in Section 5.3.2, followed by homomorphic encryption in Section 5.3.3. Section 5.3.4 is based on multi-party computation and private set intersection schemes. Cryptographic accumulators are used in Section 5.3.5 followed by zero-knowledge proof schemes in Section 5.3.6. The set membership evaluation approaches conclude with Section 5.3.7 and its attribute-based encryption schemes. The schemes based on private-pattern matching are considered additional cryptographic approaches as they mainly address different objectives, such as privacy-preserving area limitation, in Section 5.3.8. Section 5.3.9 presents a policy-based approach that gives an alternative solution to the cryptographic schemes that is based on a need-to-know evaluation. The section results are considered an optional solution to be integrated with other approaches.

Section 5.4 describes additional privacy features above the privacy layer. Section 5.5 briefly addresses the question of how to integrate the approaches into provider and service environments and databases.

<table>
<tr><td>Interface layer</td><td colspan="10">Athletes, DCO, and car sharing interfaces</td></tr>
<tr><td rowspan="2">Application and feature layer</td><td colspan="8">Whereabouts, carsharing, vehicle evaluation</td><td>Whereabouts</td><td>Whereabouts</td></tr>
<tr><td colspan="7">Point in polygon, distance measuring</td><td>Point in polygon</td><td>Need to know</td><td>Area limitation</td></tr>
<tr><td>Privacy and cryptography layer</td><td>Grid cell hashing</td><td>Probabilistic methods</td><td>Multi-party computation</td><td>Private set intersection</td><td>Zero knowledge proof</td><td>Cryptographic accumulators</td><td>Attribute-based encryption</td><td>Homomorphic encryption</td><td>Policy-based</td><td>Private pattern matching</td></tr>
<tr><td rowspan="3">Positioning and point in polygon layer</td><td colspan="6">Set membership thru parent-child evaluation</td><td>Point in rectangle thru dot product</td><td>DB-based PiP</td><td>Pattern and substring matching</td></tr>
<tr><td colspan="6">S2 and Geohash cells</td><td>Geohash cells</td><td>Long/Lat and polygons</td><td>S2 and Geohash cells</td></tr>
<tr><td colspan="10">World Geodetic System (WGS 84)</td></tr>
<tr><td>Hardware layer</td><td colspan="10">GPS, Cell-ID, WLAN, Bluetooth positioning and others</td></tr>
</table>

Figure 5.2: Concept instances

5.1.2 Privacy versus security

The following describes the different use case models and their relation to security and privacy features and understanding.

In the general security model, the message sent by Andy to Becky must be protected from an eavesdropper, i.e., an adversary such as Ellie.

$$\text{Andy} \xrightarrow{\text{message}} \text{Becky}$$

The message is protected by encryption with a mechanism that allows only Andy and Becky to read the plain text message.

$$\text{Andy} \xrightarrow{\text{enc(message)}} \text{Becky}$$

There is no eavesdropper in the general privacy model, but the message, i.e., the data, must be protected from Becky running a service.

$$\text{Andy} \underset{\text{provideservice}}{\overset{\text{data}}{\rightleftarrows}} \text{Service operated by Becky}$$

The question is, how can Andy *protect* his data and *use* Becky's service at the same time. This is a *self-directional* LBS example because the user and the target are the same entity.

$$\text{Andy} \underset{\text{provideservice}}{\overset{\text{protect(data)}}{\rightleftarrows}} \text{Service operated by Becky}$$

In a *uni-directional* LBS, the roles of the target and user are adapted by two actors, as in the doping control whereabouts use case. The DCO, Carrie, initiates the LBS and queries Andy's position.

$$\text{Andy} \overset{\text{request(position)}}{\underset{\text{protect(data)}}{\leftrightarrows}} \text{Service} \overset{\text{request(position)}}{\underset{\text{protect(data)}}{\leftrightarrows}} \text{Carrie}$$

In a *bi-directional* use case, such as a location-based social network, the users of the service exchange data that should be hidden from the service itself. The users take both roles and consume the service equally.

$$\text{Andy} \overset{\text{protect(data)}}{\underset{\text{provideservice}}{\rightleftarrows}} \text{Service} \overset{\text{protect(data)}}{\underset{\text{provideservice}}{\leftrightarrows}} \text{Carrie}$$

5.1.3 The intuitive approach

Given the roles described above, there are several attack scenarios in which a malicious party tries to obtain information that is considered private. First, the focus is on the target and the service provider. If each member of this relationship is trusted, this would mean that the target provides its actual location, and the service provider does not use it in any other way. In the case of a *naive* geofence evaluation, the service provider would provide a geofence to the target, and the target would respond with an honest answer as to whether it is *inside* the geofence or *not*.

One drawback of such an approach is that if the service needed to check against global geofences in a social network or fitness tracking scenario, this intuitive approach would require the target device to process a flood of requests. Furthermore, it is not comprehensible to trust a service provider when a second intuitive approach would send the target device's location in plain text to the service that could perform extensive computation within a spatial database, such as PostGIS. The use of target location information might be too lucrative for the service provider, such as targeted advertising [11]. Even if the service did not use it, this is a single point of attack for an adversary to obtain private information. As the news about fitness apps like Polar and Strava have shown, even location information can be indirectly leaked and associated with other data sets. In this particular case, the data revealed and confirmed the locations of military sites [104].

It is assumed that the service provider should not know where the user is located within the geofence. Therefore, the position must be cryptographically secured before sending it to the service.

Given that a service provider should not know the exact location of a target device, the intuitive approach is to send the geofence to the target device that responds with just a binary *yes* or *no* answer. This assumes that the target device and the user can be trusted.

Furthermore, the intuitive approach would not work when a service provider should not know the geofence, as the area could indicate the privacy reason why it was selected in the first place. This is the case in the whereabouts use case in the privacy-aware doping control scenario.

Germany's north-south expansion is 876km, and its west-east expansion is 632km, which results in a rectangular area of A_{Ger}=553,632km^2. Using successive location requests to a target device, each step can cut the previous area in half, Equation 5.1.

[1] Image source: Data used to produce the figure is from https://www.suche-postleitzahl.org/.

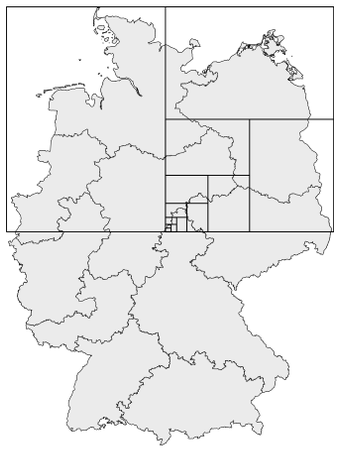


Figure 5.3: If a service subsequently requests a target position in Germany with a rectangular geofence, it would need at most 25 requests to determine the position with an accuracy of less than 200 m^2. Borders inside the geofence show the one-digit postcode areas of Germany[1]

$$A_{i+1} = \frac{A_i}{2} \tag{5.1}$$

To locate a device by querying its position against reduced geofence (rectangle) sizes, approximately 25 queries would be required to determine the location of a target to an area of less than 200m^2, Equations 5.2 and 5.3.

$$\frac{\mathrm{A_{Ger}}}{2^{25}} = 0.0165 km^2 \tag{5.2}$$

$$\sqrt{0.0165 km^2} \approx 128 m \tag{5.3}$$

5.1.4 The cryptographic approach

A five-step protocol is required to evaluate whether a position is within an arbitrarily shaped geofence without revealing it.

First, the geofence *gf* is geo encoded $ge()$, keeping its arbitrary shape.

$$gf \rightarrow ge(gf) \tag{5.4}$$

Second, the long/lat position *pos* is geo encoded $ge()$ using the same encoding system as the *gf*.

$$pos \rightarrow ge(pos) \tag{5.5}$$

Third, to keep the exact position *pos* a secret during the $ge(pos)$ evaluation phase is cryptographically encoded $enc()$. This is called *proof* and is done by the target client, i.e., the *prover*.

$$ge(pos) \rightarrow enc(ge(pos)) \tag{5.6}$$

Fourth, to evaluate an encoded position $enc(ge(pos))$ against a geofence gf, the geofence must also be encoded $enc()$. This is done in a *setup phase*, see below.

$$ge(gf) \rightarrow enc(ge(gf)) \tag{5.7}$$

In the fifth and last step, a verification function $f()$ accepts the cryptographically encoded *geofence* and *position*. It returns *true* if the position is inside the geofence, *false* otherwise. This is the *verification* performed by the service, i.e., the *verifier*.

$$f(enc(ge(gf)), enc(ge(pos))) = \begin{cases} \text{true} \\ \text{false} \end{cases} \tag{5.8}$$

Setup

Before using privacy-preserving geofence evaluation, it may be necessary to exchange data between the client and the service. The data may be created by the users, the services, or a bilateral protocol. This data can be client-encoded geofences or security keys.

For privacy-preserving distance measuring between users, the setup can be performed by all targets, see Section 5.4.1. This is a multilateral scenario since it involves the parties, the clients, the service, and a group of target users.

In the doping-control use case, the position *pos* needs to be *revealed* to the DCO and PARADISE back-end system if the athlete is not within a self-defined privacy area and the DCO is in close proximity. This behavior is formalized as follows.

$$f(enc(ge(gf)), enc(ge(pos))) = \begin{cases} \text{true} \\ \text{false, pos} \end{cases} \tag{5.9}$$

5.2 Geofence representation

The objectives of creating geofences with arbitrary shapes and supporting privacy-preserving analysis with various protocols require that a polygon based on long/lat coordinates has to be converted to a different geo encoding system. The hierarchical data structures with space-filling curves are the most suitable among the described techniques. The location grid-based encoding systems *Geohash* and *S2* were evaluated and considered the most ideal due to the construction of child and parent cells. The terms *cell* or *grid cell* refer to both individual Geohashes and S2 cells.

The hierarchical structure of these systems allows for an efficient point in polygon computation. The different cell sizes from one resolution level to the next permit variable accuracy levels for any arbitrarily shaped geofences adapted to the general use cases and privacy methods employed. As shown in Figures 5.4 and 5.5, the process results in a set of identifiers, each describing a sub-region of the geofence. By increasing the number of cells used for the encoding, i.e., modifying the minimum and maximum resolution level, the approximation can be adjusted [121].

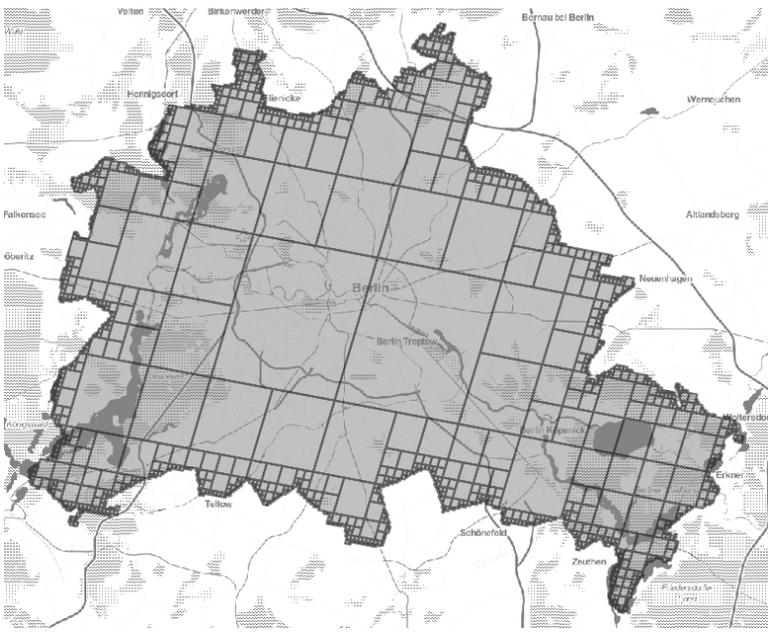


Figure 5.4: The Berlin geofence is represented by normalized S2 grid cells using various levels.[2]

Figure 5.5: Hierarchical S2 cell levels. Cell A (level 13) can be hierarchically reduced to cell B and C (levels 12 and 11). Highlighted is the S2 token representation corresponding to Table 5.1, where the hierarchy can be seen in binary form.

Table 5.1 shows the three S2 cell values from Figure 5.5 in binary form and their respective cell level.

5.2.1 Normalized representation

In a *non-normalized* form, the long/lat geofence is represented by Geohashes and S2 cells of the same length or plane. Using large cells to define a geofence, the area measured

[2]Image sources: Figures 5.4 and 5.5 are from the original publication, Victor and Zickau (2018), and were

Table 5.1: S2 cell hierarchy

In Fig. 5.5	S2 cell in binary	S2 cell level
A	0100 0111 1010 1000 0101 0001 1010 1100 ...	13
B	0100 0111 1010 1000 0101 0001 1011 0000 ...	12
C	0100 0111 1010 1000 0101 0001 1100 0000 ...	11

in square meters will mainly differ from the original polygon. However, the advantage is that the number of cells can be relatively small. By using small cells to represent the geofence, the area comparison between the polygon and its cell representation can be arbitrarily accurate, see Section 4.2. The disadvantage is that the number of cells increases for a more accurate representation. Figure 5.6 illustrates an example using 90 Geohashes to represent a very *rough* shape of Berlin.

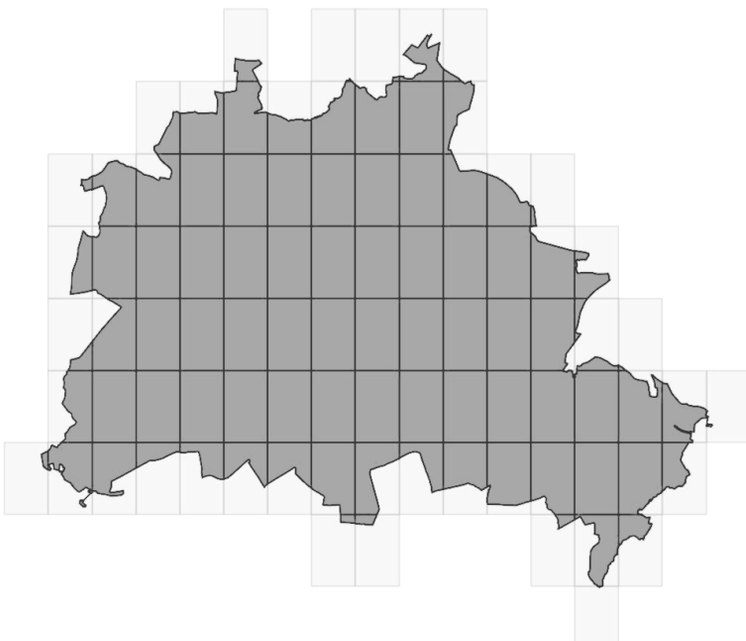


Figure 5.6: The geofence in the Berlin area, represented by 90 non-normalized Geohashes.

In the *normalized* form of a cell representation, child cells are represented by their parent cells. In the case of S2 cells, four children are represented by one parent cell. A Geohash parent consists of 32 children. The advantage of the normalized form is that fewer cells are needed, and the geofence representation is more accurate. Figure 5.7 shows 815 normalized Geohashes to represent a more *fine-grained* shape of Berlin.

For the use of S2 points and S2 cells (parents) in privacy and security use cases, the following properties (which also apply to Geohashes) are used. Each parent of an S2 point and its subsequent parents always cover the S2 point, i.e., a point or child cell is always inside all its parents, see Equation 5.10.

$$\forall \text{point}(\textit{is}_{\text{parent}}(\text{point}) \rightarrow \textit{inside}_{\text{parent}}(\text{point})) \tag{5.10}$$

Figure 5.8 presents a simplified version of an S2 point and four of its parents. The S2

collaboratively created by Friedhelm Victor and Sebastian Zickau.

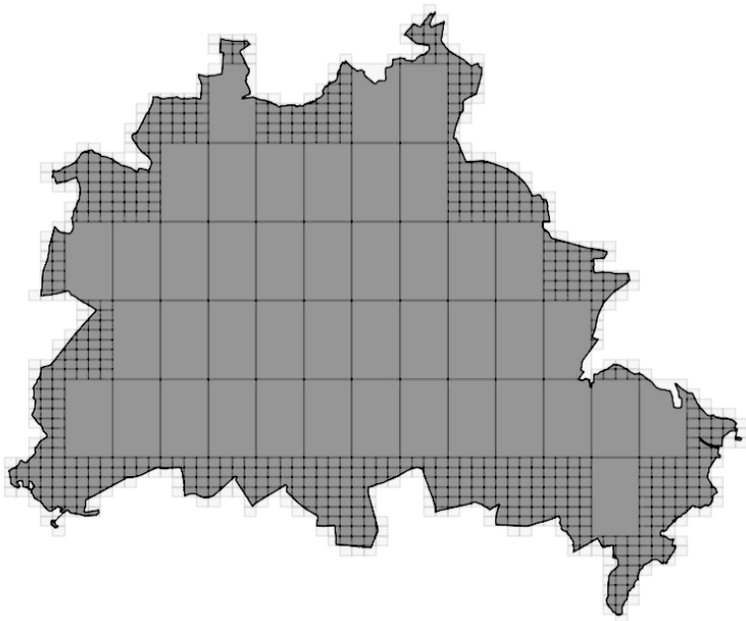


Figure 5.7: The geofence in the Berlin area is represented by 815 normalized Geohashes.

values have been modified (4444 , 4440 , 4400 , 4000 , 40000) to clarify their use in the privacy-preserving schemes and protocols. 4444 represents an S2 point and four of its parent values (4440 , 4400 , 4000 , 40000).

Figure 5.8: The values of the geocoded position as simplified S2 point 4444 and its four direct parents (simplified as 4440,4400,4000,40000).

5.2.2 Inverse geofences

An *inverse* geofence, or *negative*, *inverted*, or *inside out* geofence, covers the outer area of a given polygon. As an example, consider a polygon representing the borders of a country. The complete area outside the country is defined by its inverse geofence, i.e., it covers the area of the entire globe but not this specifically given polygon, see Figure 5.9 for a simplified example.

Figure 5.10 shows Berlin represented by normalized S2 grid cells and by its *inverted* representation.

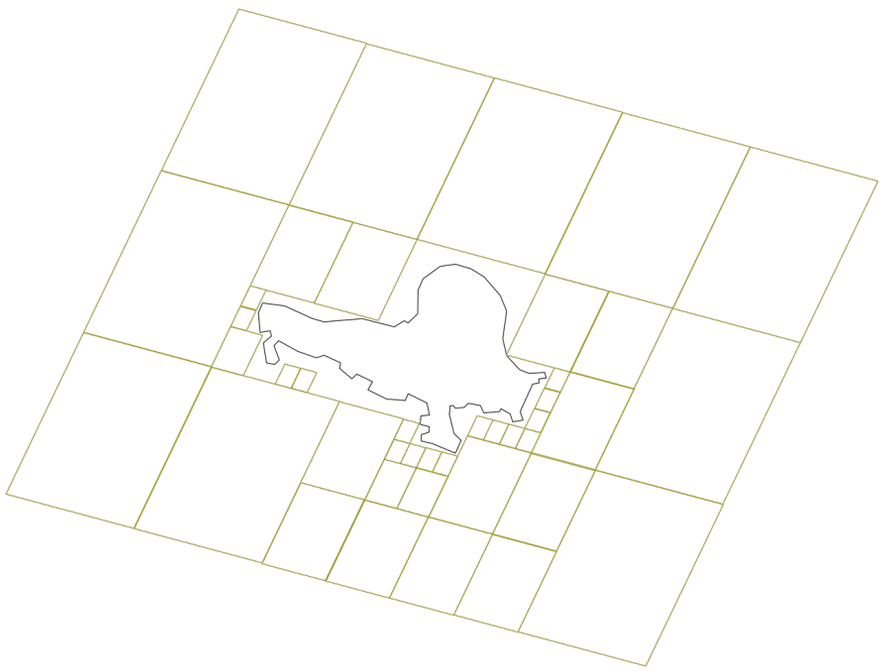
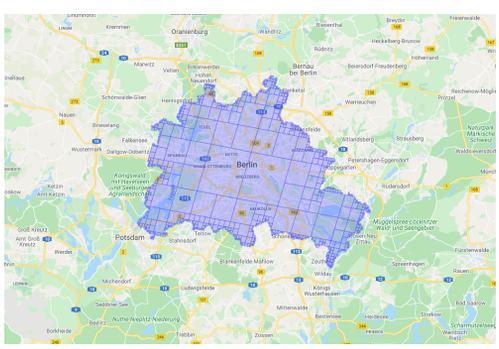


Figure 5.9: Example of an arbitrarily shaped geofence represented by its inverted S2 grid cells surrounding the geofence. A minimum cell level in this example is set. That is why the S2 area ends eventually. The examples for inverted geofences described in the thesis usually cover the whole area around a smaller geofence, i.e., it almost covers the entire globe. Therefore five of six S2 face values at level 0 are used.



(a) The Berlin area represented by its 810 normalized S2 grid cells using different levels.

(b) The same geofence of the Berlin area represented by its 896 inverted and normalized S2 grid cells also using different levels.

Figure 5.10: A normalized S2 representation of Berlin and its inverted form for the same underlying polygon.[3]

Section 7.3.1.2 of the evaluation compares the number of grid cells needed to represent different areas using inverted geofences. Using the S2 representation, almost the same number of cells is required to represent complex polygons in the normalized model and its inverted counterpart. In the Berlin example above, 810 cells are used in the normalized form and 896 in its inverted variant.

5.3 Privacy-preserving methods

The methods and technologies for achieving privacy-preserving geofencing are listed in the table below. They can be divided into *hashing, probabilistic methods, homomorphic encryption, multi-party computation, private set intersection, zero-knowledge proofs, cryptographic accumulators, attribute-based encryption, private pattern matching*, and *policy-based approaches*. Table 5.2 shows these approaches along with the objectives summarized in Section 2.6. The goal is to address all objectives using the ten approaches. As shown in Figure 5.2, these approaches are listed in the *privacy and cryptography layer*. Each subsection from 5.3.1 to 5.3.9 is dedicated to a single (cryptographic) approach. The results are shown in Table 7.2.

[3]Image sources: Underlying map data from Google Maps 2018 GeoBasis-DE/BKG (2009), Google. Figures were created using the QGIS open source application.

Table 5.2: Cryptographic approaches and objectives

	1: Data send to service	2: Precise target position	3: Fast encoding	4: Service learns what is necessary	5: Geofence not known to service	6: Service may limit geofences	7: Privacy-preserving distance measuring	8: Position shown if outside	9: Arbitrary shaped geofence	10: Feasible memory sizes	11: Geofence updates not necessary (privacy)	12: Low-powered device	13: Geofences may be added, deleted, or updated	14: On-the-fly geofences	15: Separation of positioning and encoding	16: Services do not get shared secrets	17: Support Bluetooth, Wi-Fi, etc.
1 Grid cell hashing	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
2 Probabilistic methods	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
3 Homomorphic encryption	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
4 Multi-party computation	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
5 Private set intersection	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
6 Zero-knowledge proofs	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
7 Cryptographic accumulators	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
8 Attribute-based encryption	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
9 Private pattern matching	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
10 Policy-based approaches	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

5.3.1 Grid cell hashing

The representation of a geofence replaced by Geohashes and S2 cells allows the conversion of these strings or binary code into another type of data. Converting a string value, such as a secret password, into its hashed value allows private proof of knowledge. Cryptographic hashing primitives and definitions are presented in Section 4.4.1. The password cannot usually be retrieved by knowing only its hashed value. It can be considered a *non-reversible encryption*, i.e., a *one-way* function. A way to use this in privacy-preserving geofences is by securely hashing each value of the grid cell representation.

1. Geo encode polygon of the geofence.
2. Generate grid cell values of the polygon.
3. Hash all grid cell values.
4. Store hashes in a database.

The target user can convert a geofence into a list of hash values in a *setup* phase, similar to the way passwords are processed. In this case, the geofence is not known to the service provider. This property addresses the LBS class attribute *only the client knows the geofence* as described in Table 3.2.

The mechanisms for evaluating a given position against these hashed values are performed as follows. A mobile device positions itself, learning its long/lat position. This position is converted to its S2 cell, using the S2 point value (level 30), together with all parent S2 cells (levels 29-0). All 31 values are hashed on the mobile device and sent to the service provider. The hashed S2 grid cells are compared with the list of hashes of the geofence. If there is a match, the mobile device's location is within the geofence. If there is no match, the mobile device is outside the geofence.

1. Calculate long/lat values of the current position.
2. Find the S2 point for the position.
3. Generate all S2 parents of S2 point.
4. Generate all 30 hash values of S2 parents.
5. Send hash values to the service.
6. Compare hash values with the list of geofence hashes in the database.
7. If there is a match, the target is within the border of the geofence, if not, the target is outside.

5.3.1.1 Simplified example

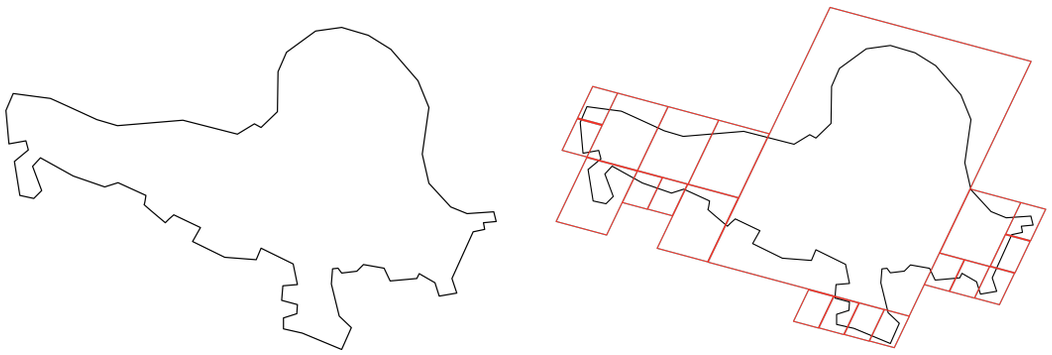
In Figure 5.11, an arbitrarily shaped geofence *gf* is shown. The example postcode area is the district Charlottenburg in Berlin. The geofence is represented by 20 cells.

The set S of S2 cells c_n representing the geofence is defined as

$$S_{gf} = [c_1, ..., c_n]. \tag{5.11}$$

These values are encoded by applying a cryptographic hash function $h()$. The set of hashed values H is defined as

$$H_{gf} = [h(c_1), ..., h(c_n)]. \tag{5.12}$$



(a) An arbitrary shaped polygon as an example geofence.

(b) The example geofence geo encoded with simplified S2 cells.

Figure 5.11: An arbitrarily shaped geofence polygon and its S2 cell representation with different levels (simplified) as the basis for the examples.

As shown in Section 5.2, the geo encoded position $ge(pos)$ as an S2 point and its *parents* are defined as

$$S_{pos} = [ge(pos), parent_1(ge(pos)), ..., parent_n(ge(pos))]. \tag{5.13}$$

In general, a parent of a geo encoded position is an S2 cell c with a level x and is defined as

$$parent_x(ge(pos)) = c_x. \tag{5.14}$$

The cryptographically encoded set H_{pos} is defined as

$$H_{pos} = [h_{pos}(ge(pos)), h_1(parent_1(ge(pos))), ..., h_n(parent_n(ge(pos)))]. \tag{5.15}$$

The evaluation function $f()$, i.e., a set membership function, considers both sets and yields a *true* or *false* value.

$$f(H_{gf}, H_{pos}) \rightarrow \begin{cases} \text{true} \\ \text{false} \end{cases} \tag{5.16}$$

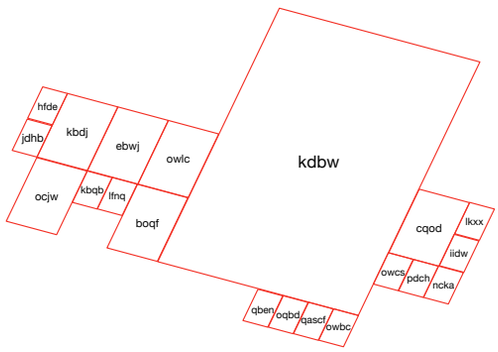
In Equation 5.10, it was proven that for any S2 point, all its 30 parents include the initial point. Equation 5.17 describes that if a point is inside the geofence boundaries, one of its parents is a set member of S_{gf}.

$$\forall pos \exists parent_{pos}(inside_{gf}(pos) \rightarrow parent_{pos} \in S_{gf}) \tag{5.17}$$

A derivation of this equation also provides us with the characteristics that for any S2 point outside the geofence S_{gf} none of its parents will be included in the set (Equation 5.18). $outside_{gf}(x)$ describes any position which is outside the geofence.

$$\forall pos \nexists parent_{pos}(outside_{gf}(pos) \rightarrow parent_{pos} \in S_{gf}) \tag{5.18}$$

(a) Simplified numeric S2 cell value examples with different levels indicated by the 0s.

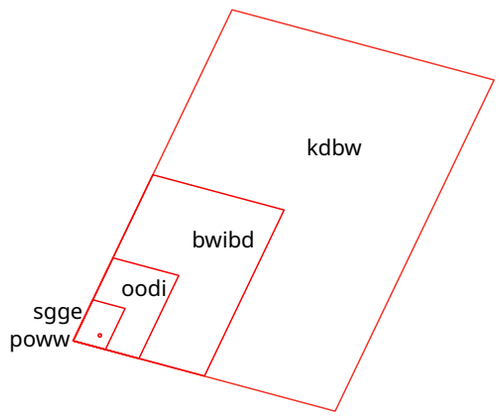


(b) The hashed strings of the S2 cell values of (a) in a simplified form.

Figure 5.12: The cryptographically hashed values (b) of the numeric cells (a) of the geofence example S2.

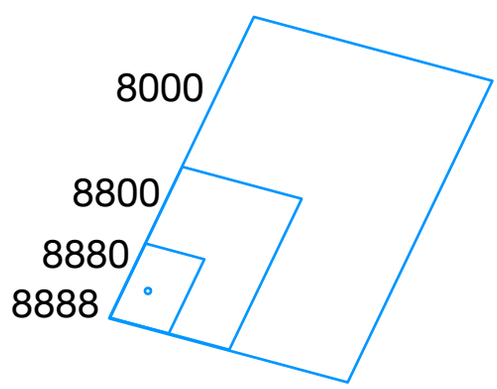
Figures 5.13 to 5.15 exemplify this graphically.

(a) An S2 point (4444) and four of its parents (4440, 4400, 4000, and 400000).

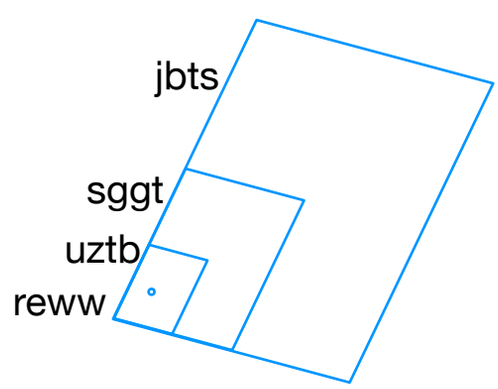


(b) Simplified hash values of the point and parent values.

Figure 5.13: An S2 point and its four parents (a) and their cryptographically encoded hash values (b). The point 4444 is located inside the geofence example.



(a) An S2 point example (8888) and its three parents (8880, 8800, and 8000).



(b) Simplified hash values of the S2 point and its parents.

Figure 5.14: An S2 point and three parents (a) and their cryptographically encoded hash values (b). The point 8888 is outside of the geofence example.

In the simplified example, the four hashed parent values of *pos* evaluate to

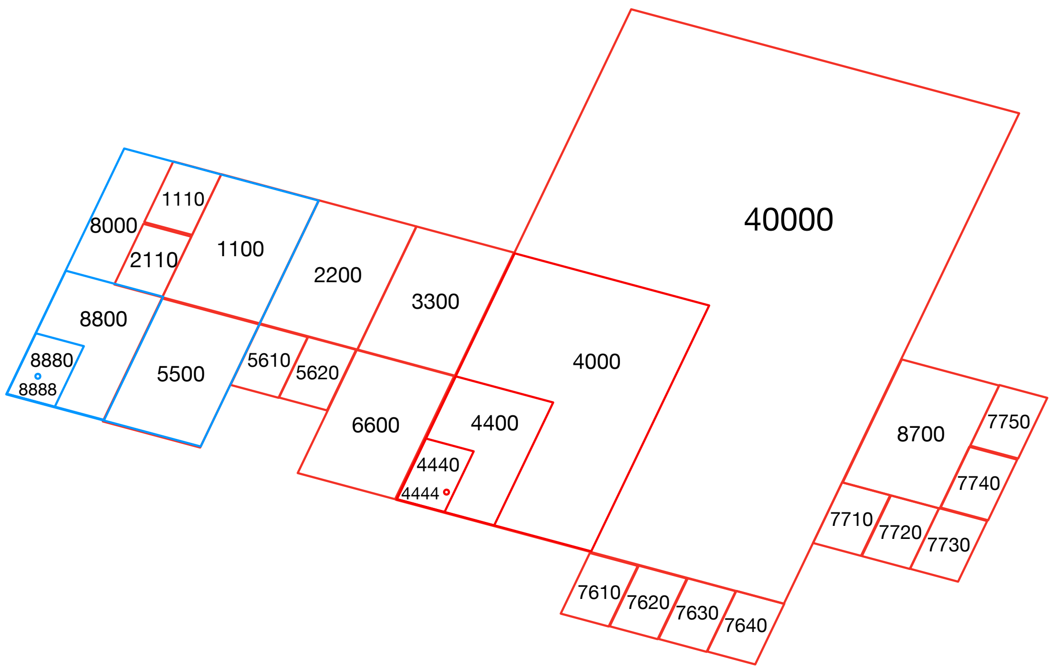


Figure 5.15: Both the geocoded geofence and the outer (blue areas) and inner (red areas) positions and their respective parents. The parent of an outer position is never part of the geofence set. A parent of an inner position is always part of the geofence set.

$$
\begin{aligned}
&\vdots \\
f(\texttt{kdbw,sgge}) &\rightarrow \text{false} \\
f(\texttt{kdbw,oodi}) &\rightarrow \text{false} \\
f(\texttt{kdbw,bwibd}) &\rightarrow \text{false} \\
f(\texttt{kdbw,kdbw}) &\rightarrow \text{true} \\
&\vdots
\end{aligned}
$$

It must be noted that this *basic* version of hash values cannot be considered optimal for privacy requirements. The S2 point and S2 cell values are finite and, unlike passwords, known in advance. The message space is very small so that hashed S2 cells can be generated in advance using the S2 library. Hashing all values would only require reasonable computational power since hashing is computationally cheap. Also, cells can be omitted, such as oceans or countries where the service provider does not operate, e.g., a carsharing company that operates only in a few cities. The following subsections address these shortcomings by adding cryptographic primitives to the hashing mechanism.

5.3.1.2 Salt, pepper, and nonce

Hash functions have a *deterministic* nature. Within the GCH approach, this is the desired property to compare the parents to the stored values of a geofence. Nevertheless, since the values are short and known to an attacker, they must be *salted* with a *random* value, i.e., a *secret*, for both the geofences grid cells during the setup and the position and its parent values during the service usage.

This is solved by introducing *salt* values before hashing each S2 cell. This means that the S2 cell value is combined with an additional string, the *salt*, before hashing. An

attacker would not be able to generate all hashes for all S2 cells as the *salt* is not known, comparable to an unknown password. The sequence diagram in Figure 5.16 shows the two options in detail.

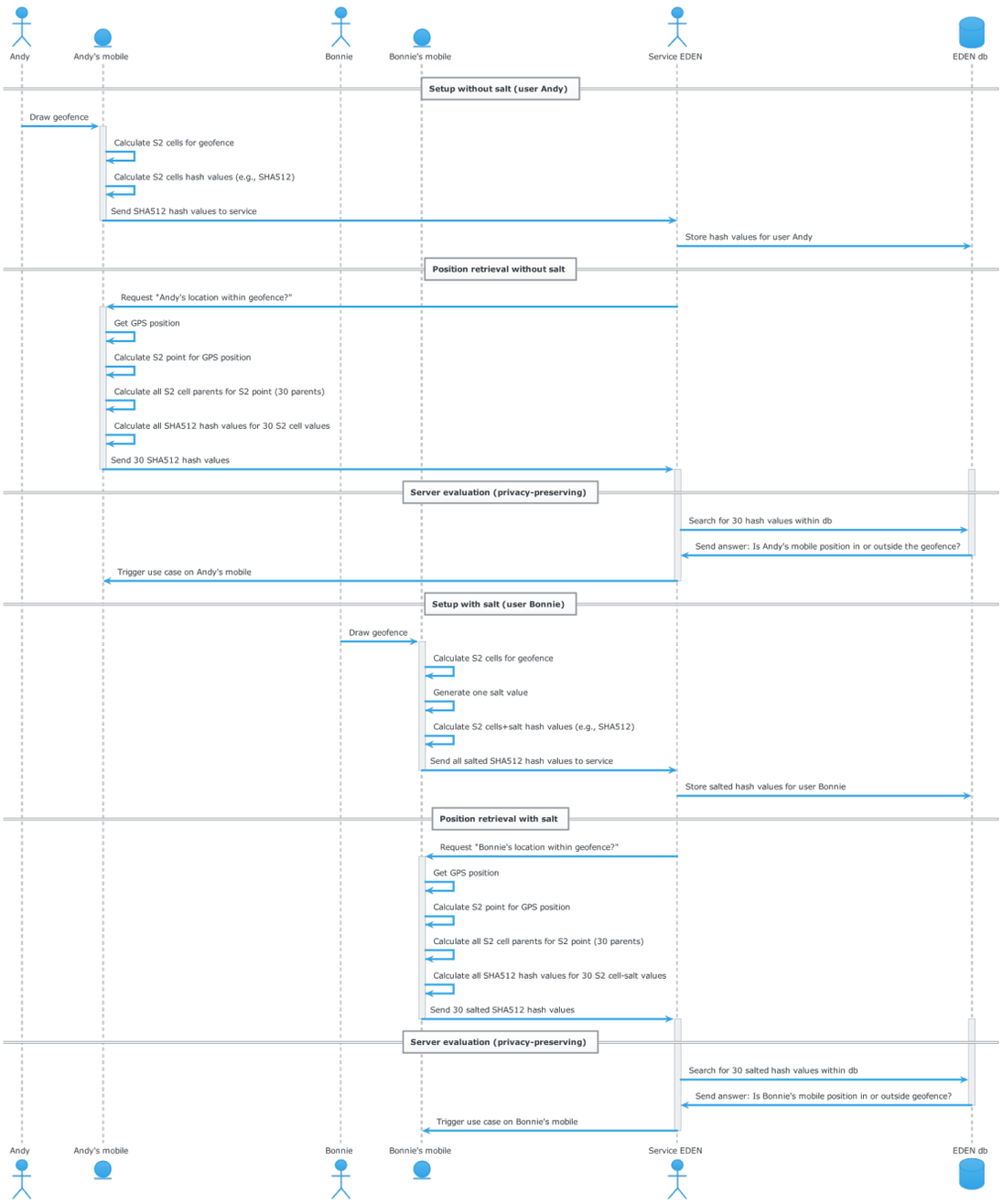


Figure 5.16: Grid cell hashing sequence diagram. The diagram contains two approaches to the hashing protocol, one with hashing only (user Andy) and the other with a salt value before hashing (user Bonnie). The three phases of each variant are divided into a setup phase, a position retrieval phase, and a server evaluation phase. A service called EDEN and its database are used as examples.

In the context of password storage, the salt value is a *non-secret* value stored alongside the hash. A password salt prevents attacks with prefabricated data, such as *lookup tables*, *reverse lookup tables*, and *rainbow tables*. In addition to a *salt*, other terms for *secret* values are used in applied cryptography, e.g., *pepper*, *nonce*, and *garlic* (see below).

To make it impossible for an attacker to create a lookup table, the salt should be the same size as the hash function's output. For example, 256 bits (32 bytes) with SHA256, so the salt should be at least 32 *random* bytes [90]. Example values for all steps can be found in

Table 5.3.

Table 5.3: Example cell values, salt values, and hashes

Cell values *0100011110101000010100001110000111111100000000000000000000000000* *5163465907594330112*
Salt values **d122f6531d28c6f5** **bc1c81f4898944bb**
Values to be hashed *0100011110101000010100001110000111111100000000000000000000000000***d122f6531d28c6f5** *5163465907594330112***bc1c81f4898944bb**
Hash values = SHA256(*cell value* + **salt value**) 2B1C1E93A1B0F7168074A7DFB14D681CB3221D76F8217F2401755BE16075F010 9B40E8E2BDD2390B0DF0BFD6A824B2BFDA54B7039AD5C2BAF91EA46135FCCA31

Pepper Compared to a *salt* used to secure passwords in the storage of a service's database, where the plaintext salt is stored, a secret *pepper* value must not be stored with the output of a hash function. A pepper value must be secured so that it remains secret to a client or application. The NIST recommends using pepper values of at least 112 bits [201]. In the document, it is not named pepper but described as "... an additional iteration of a key derivation function using a salt value that is secret". The pepper protects against weak passwords. Our grid cell values can be considered very weak in the presented context. The encryption equivalent to a pepper value is a secret key. As long as the pepper is secret, the hashed grid cell values are also confidential. Using a pepper value also protects against *dictionary attacks* and *brute-force attacks*.

When using salt/pepper values within the privacy-preserving distance measurement approach in Section 5.4.1, it is necessary to evaluate which security feature should be used to protect the locations of more than one user. The use of a *pseudorandom generator* also needs to be discussed and evaluated.

Time-variant nonce A *nonce* value adds *originality* to a particular message. It can also be generated based on a time-variant. The NIST definition of a nonce value is "*A time-varying value that has at most a negligible chance of repeating, for example, a random value that is generated anew for each use, a timestamp, a sequence number, or some combination of these*" [91].

5.3.1.3 Hash-based message authentication code

Following the introduction of MACs in Section 4.4.1.2, a *hash-based message authentication code* (HMAC) as described in [167] is a special type of a *message authentication code* (MAC), also referred to as a *tag*. MACs can be derived from block ciphers or hash functions [202]. The standard function is *HMAC* [167], which is applied to a hash function $h()$, which is derived from a compression function $f()$. Equation 5.19 shows how the keyed hash value HMAC of a message m is calculated.

$$HMAC(k, m) := h((k \oplus opad)||h((k \oplus ipad)||m)) \tag{5.19}$$

The inner and outer padding, ipad and opad, are performed with two fixed and distinct strings

$$\text{ipad} := \text{the byte 0x36 repeated } r \text{ times},$$
$$\text{opad} := \text{the byte 0x5C repeated } r \text{ times}.$$

The hash function $h()$ is applied twice to prevent the message from being modified. Without it, the so-called *length extension problem of iterating hash functions attack* could be applied [90]. As an aside, using block ciphers to construct MACs is another way to ensure the authentication and integrity of a message. CBC-MAC is one of the most common and essential block-cipher-based algorithms [203] [202].

Time-based One-time password The HMAC-based *One-Time Password algorithm* (HOTP) is described in RFC 4226 (2005) [204], and the evolutionary *Time-based One-Time Password algorithm* (TOTP) is described in RFC 6238 (2011) [205]. The two algorithms are proposed and used in the realm of two-factor authentication mechanisms, such as the Google Authenticator [206]. The algorithms are event-based and can be implemented in low-cost devices with little computational capabilities, such as smart cards, USB sticks, and SIM cards.

The algorithms use a *shared secret K*, such as the salt or pepper values mentioned above. In RFC 4226, this is set to a minimum value of 128 bits, although 160 bits is recommended. They also employ an 8-byte so-called *moving factor* or *counter value C*. In the user or device authentication context, they also include the notion of a *throttling* parameter *T* and *resynchronization* parameter *s*. The HTOP algorithm is based on an increasing counter value *C*, a static symmetric key *K*, known only to the client and validation service, and a cryptographic hash function $H()$ [204]. The algorithm uses the $\text{HMAC}_{\text{SHA1}}$ computation truncated with a user-supplied *simple enough* value, see Equation 5.20. This converts the HMAC output to an HTOP value, see Equation 5.21. *d* is the least significant base-10 digits of HTOP.

$$HTOP(K, C) = Truncate(HMAC_H(K, C)) \tag{5.20}$$

$$HTOPvalue = HTOP(K, C) \bmod 10^d \tag{5.21}$$

To support a time-based moving factor, TOTP was derived from the HTOP algorithm, which uses a *time- and time-step referenced value T* as the *counter value C*, for example, a Unix time epoch minute value, i.e., the number of seconds elapsed since midnight UTC of January 1, 1970. TOTP utilized HMAC-SHA-256 and HMAC-SHA-512.

The following terms are applied in the TOTP algorithm, *X* stands for the time steps in seconds, the default is 30. T_0 is the Unix time for the step count; the default is 0, i.e., the Unix epoch. It is also a system parameter. The basic definition of TOTP is $HTOP(K, T)$, where *T* is an integer representing the steps between T_0 and the current Unix time, i.e., the counter C_T, see Equations 5.22 and 5.23.

$$TOTPvalue(K) = HTOPvalue(K, C_T) \tag{5.22}$$

$$C_T = T - T_0/T_X \tag{5.23}$$

The values sent to the service must be independent of those previously sent. The service must not receive information about which values are in a parent-child relationship and that a position has not changed or has changed only slightly over time. Otherwise, this could give away the target's speed, for example.

To satisfy this requirement, a unique time-based value is added to the grid cell values, including a secret, in the manner of an HMAC-based One-time Password algorithm (HTOP) or similar to the Time-based One-Time Password algorithm (TOTP). The second requirement demands that the initial values for the geofence be passed to the service. The service is only allowed to compute, i.e., know, new values based on the TOTP algorithm only for the given cells within the geofence representation. The service *must not* know the initial values (hash *plus* secret) of child or parent values that are not part of the geofence representation. To formalize this, the following nine steps are performed.

1. Generation of geofence values by the client during setup.

$$GF_x = hash(cell_x + secret) \tag{5.24}$$

2. Sending these values to the service and storing them in a database is also part of the setup.
3. Generation of the parents by the client during the evaluation.

$$pos_x \text{ AND } parents_x \tag{5.25}$$

4. Hashing all these values together with the target's particular secret.

$$h(parents_{pos} + secret) \tag{5.26}$$

5. Calculate the Unix time T and its current epoch T_X, e.g., in one-minute intervals, to obtain the counter value C_T.
6. Hashing all these values.

$$h((h(parents_{pos})) + C_T) \tag{5.27}$$

7. Send all these values mixed up to the service.
8. Construct the hashes with the same interval values for all values in the geofence C_T.

$$h(GF_x) = h((h(cell_x + secret)) + C_T) \tag{5.28}$$

9. Compare the values from the client with any newly (currently) hashed values. If there is a common value, the client is inside the geofence.

The changing values for C_T can be pre-compiled by the service to save time for calculating, e.g., the number of geofence values. Alternatively, the value C_T can be sent by the client to the encoded parent values to make it independent from the Unix time epoch or non-computable. Figure 5.17 shows a sequence diagram of the steps described above. Along with Figure 5.18 where step-by-step tables present an example with simplified values for

the grid cells, geofence, and hashes.

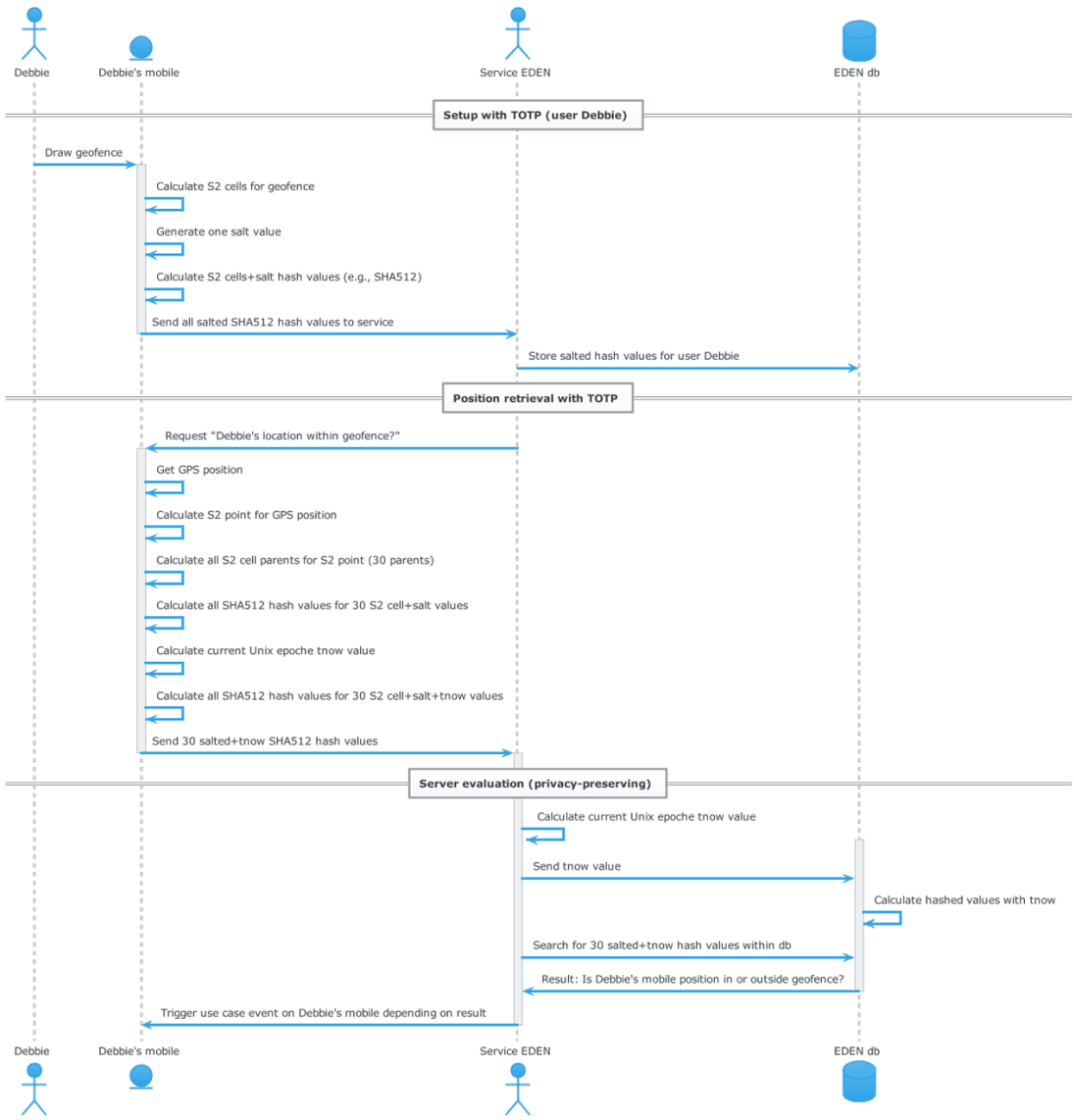


Figure 5.17: Sequence diagram for the grid cell hashing scheme along with a time-based one-time password (TOTP) algorithm that uses a Unix epoch time value and a salt value before hashing.

5.3.1.4 Key derivation functions

A *key derivation function* (KDF), as the name suggests, derives one or more secrets from a known (secret) value, e.g., a master key or a password, utilizing a pseudorandom number generator. KDFs can be used to derive secrets in a specific format or length. For example, a group element could be used to derive an AES key in a Diffie-Hellman key exchange [207]. Keyed cryptographic hash functions, also called HMACs, can be used as random functions to derive keys. The term *password-scrambling* is also used in KDFs, particularly for password hashing.

After some prominent password breaches, e.g., Playstation Network in 2011 with the information leak of 77 million accounts [208], the NIST Password Hashing Competition between 2013 and 2015 addressed the importance of reliable and secure hashing algorithms, especially for user passwords [209]. The winner of this competition was the *Argon2* KDF, designed by Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich from

parent values			
time	t0	t1	t2
cell values of target client	1000	1000	1010
	1100	1100	0010
	1101	1111	0001
hashed cell values	A3FT	A3FT	ZGHT
	HH78	HH78	OP21
	QWEC	TG54	OOI9
hashed cell values + salt	87HG	87HG	ZH99
	44ZT	44ZT	FTR6
	7ASY	XCV3	OOI9
hashed cell values + salt + time	54FD	ZT66	99UH
	87HG	77KJ	98LL
	665J	341T	IKKL

geofence		
t0	t1	t2
1100	1100	1100
1110	1110	1110
HH78	HH78	HH78
KL9R	KL9R	KL9R
44ZT	44ZT	44ZT
IIU8	IIU8	IIU8
87HG	77KJ	876G
OK7K	341T	11LU

Figure 5.18: Simplified example of GCH in conjunction with TOTP. The left table shows the parent values of the target at time points t0, t1, and t2. Three parent values are shown for each time point below the other. The target has two common parent values at t0 and t1. At time t2, the three values are different. The right table shows two geofence values (1100, 1110) of the service that are the same for all three times. The target is inside the geofence at time points t0 and t1 because the parent value 1100 is the same as the first geofence value. At time t2, the target is outside the geofence because no parent value matches either value. The bottom three rows of the table show S2 cell values successively hashed using different hashing methods, i.e., hashing, hashing plus salt, and hashing plus salt and time-variant. Only the last approach hides the target's information in the same parent cell 1000 (gray) in t0 and t1. However, all rows show that the target is within geofence cell 1100 at times t0 and t1 (red). Each parent value in the bottom table's last column is different, and the service cannot detect the relationship between parent and child cells.

the University of Luxembourg [210]. The contest was also used to find alternatives for weak algorithms such as *md5crypt* on older BSD and Linux systems and *NTLMv1* on older Windows systems. A common algorithm besides the mentioned Argon2 is *PBKDF2* [211] as a KDF in WPA2, OpenOffice, and WinZip. It is also used as a password scrambler in macOS or LastPass [212]. Also worth mentioning are *bcrypt* and *sha512crypt*. The former is used in OpenBSD systems, and the latter is used in Ubuntu and Debian Linux systems. A password scrambler designed to occupy large amounts of memory to reduce attack vectors is *script*. A comparison table can be found in [213], see Section 5.3.1.7.

Passwords and passphrases often cannot be used as cryptographic keys because they do not meet the required length or lack other requirements. The technique of key stretching can be used to expand the initial password to match the desired output. This can be done in a straightforward approach, Equation 5.29,

$$DK = KDF(key, salt, iterations) \tag{5.29}$$

with

- *DK* is the derived key,
- *KDF* is the key derivation function,
- *key* is the initial password,
- *salt* is a random number, and
- *iterations* is the number of iterations of a subfunction.

The *derived key* (*DK*) is used instead of the initial key or password. As part of the user's authentication, the salt and the number of iterations are stored in plain text and the hashed *key*, i.e., the password. The number of iterations prevents a brute-force attack, and the usage of the salt value prevents the use of pre-computed lookup dictionaries or rainbow tables.

The *password-based key derivation function 2* (PBKDF2) has five input parameters [214]

$$DK = PBKDF2(PRF, password, salt, c, dkLen) \tag{5.30}$$

with

- *PRF* is a pseudorandom function of two parameters and an output of length *hLen*, e.g., a keyed HMAC,
- *password* is the password from which the *derived key* (*DK*) is generated,
- *salt* is a cryptographic salt as a sequence of bits,
- *c* is the number of iterations, and
- *dkLen* is the desired key length of the *DK*.

For a comparable number of iterations, iOS 3 used 2,000, iOS 4 used 10,000, LastPass used 5,000 for clients and 100,000 for server-side hashing in 2011 [215] [216].

The NIST recommends a salt length of 128 bits [217]. As in the case of GCH, this length may need to be extended to protect it from the current power of computing hardware performance. Again, it must be emphasized that in the GCH approach, which is based on S2 values or Geohashes, the initial values are known.

Argon2 was selected as the winner of the NIST Password Hashing Competition in July

2015 [218]. Argon2 is released under the Creative Commons license and the Apache License 2.0. It is available in three different flavors

- Argon2d to resist GPU cracking attacks,
- Argon2i to resist side-channel attacks, and
- Argon2id is a hybrid version of the above and is recommended to be used.

The three flavors allow you to specify parameters that control the following

- Execution time,
- Memory required, and
- The degree of parallelism.

In the evaluation Section 7.3.2.1, the computational times and memory requirements of Geohash and S2 grid cells are compared and analyzed for different hash protocols, including SHA512, Argon2, and PBKDF2.

5.3.1.5 Inferred information

As an attacker, the service provider Becky is trying to collect information about the LBS user Andy by using all information she can get.

In the case of GCH, as described above, the additional information Becky may or could obtain is as follows.

- The *time* at which Andy sends position information (update).
- The *time interval* between two position updates.
- The *frequency* of position updates.
- In some use cases, perhaps Andy's *start* or *endpoint location* could reveal additional information (→ *known whereabouts* attack).
- In addition, the service may know which locations the target frequently "visits" if the same grid cell values occur tied to a specific time and date (→ *stationary user*, *known whereabouts*, *campus* attacks).
- The road network and routes (→ *map-based* and *mobility pattern* attacks).
- The mask cell values of the geofence where the target is located, even if the service does not know the area.
- The frequency of how often the value of such masked cell changes. This is particularly crucial when areas on the edge of a geofence consist of tiny cells in normalized form.
- With the derived information about cell sizes and frequencies, Becky could tell *how fast* the target is moving and thus what *mode of transport* it is using or whether it is moving at all and perhaps staying at its last location (→ *mobility pattern* attack).
- The use of additional information, e.g., traffic information about congestions in certain areas, could be associated with the target's location and speed (→ *map-based* and *known whereabouts* attacks).

The question is how to avoid such derived information in the GCH approach. In summary, the semi-honest party approach is considered the security model where a service uses all available data during the protocol execution.

It is also necessary to investigate whether and which of these attack vectors are also possible in other approaches that do not introduce a trusted third party.

An optimal solution also includes a countermeasure to the server's ability to associate values coming from the client. The server can detect if a client is in an area at a particular time where it was previously or if a client does not move between a position update. The desired solution should avoid collecting this information on the client-side, i.e., support for *temporal unlinkability*.

5.3.1.6 Countermeasures

The following is a list of possible countermeasures to these attack vectors.

- Using inverse geofences.
- Using non-normalized geofences. Cell sizes are dependent on the use case.
- Regularly generate new hash values for the geofence(s).
- Use of different interval times for position updates.

The following describes how these countermeasures address the inferred information attacks described in the last subsection.

Inverse geofences One approach to this problem is to use an inverted geofence at the service and evaluate the target's position against it. If the target is not inside the inverted geofence, the target is inside the actual geofence, and no information is revealed about the subarea.

Non-normalized geofences Another approach is to use the non-normalized version of a geofence to make each grid cell area within the geofence equal in size. With this approach, the service may not be able to infer whether a target is at the edge of the geofence, whereas in the normalized case, grid cell changes are much more frequent because the area of each cell is much smaller. Using a fixed level for all cells representing the geofence would cease this inferred information. Still, using the timing of the updates and the cell size as additional knowledge would let the service know when a cell change occurs and reveal, for example, at what speed the target is moving.

Regular updates of geofence initial values To overcome the problem, after some time, the service can collect information about the parent values to a particular cell or know the places that the target visits. The setup phase of geofences computed by the client could be done periodically to counteract inference by the service. In addition, temporal unlinkability can be supported by making the positioning updates infrequent. The service cannot correlate past hashed cell information with current ones using this measure. It needs to be evaluated in which use cases and how these generations need to be performed.

Alter frequency of position updates To get a handle on the inferred target velocity, the position update intervals could be randomized to make it harder for the service to gather additional information about the state of the target.

5.3.1.7 Garlic and the Catena framework

As an outlook to strengthen the values of the proposed GCH methods, the term *garlic* is introduced concerning its origin. In the publication describing the password-scrambling

framework Catena [213], Forler et al. introduce the term *garlic* as a security parameter. The framework uses a graph-based structure that depends on the number of input nodes of the permutation graph. This approach aims to make parallel password checks that use the same memory inefficient for an adversary. The garlic value defines the number of input nodes $G = 2^g$. It is designed to be adaptable to the current generation of hardware. In the paper, the authors propose to use the values $g_p = 17$ for password hashing and $g_k = 20$ for key derivation along with a 16-byte salt. The Catena framework was one of the finalists in the NIST Password Hashing Competition, mentioned in a previous section. Other finalists include the schemes Lyra2 [219], yescrypt [220], and Makwa [221].

Client-independent updates Let *gcv* be a *grid cell value* of a client-generated grid cell-based geofence and h be the hash of this specific *gcv* [213].

$$h \leftarrow Catena(gcv, t, s, g) \tag{5.31}$$

Where the parameters t, s, and g represent the values: *tweak*, *salt*, and *garlic*.

The parameter t is a multi-byte value denoted by the concatenation of

$$t \leftarrow d||\lambda||n|||s|||H(AD) \tag{5.32}$$

where the bytes indicate to

- **First byte**: $d =$ domain, 0 for password scrambler, 1 for key-derivation function, and 2 for proof of work,
- **Second byte**: λ defines together with garlic g the security parameters for Catena,
- **Third and fourth byte**: 16-bit value n the output length of the underlying hash function H,
- **A 32-bit value**: $|s|$ denotes to the total length of the salt in bits,
- **n-bit value**: $H(AD)$ is a hash of the associated data AD, which can be information such as hostnames, userIDs, company names, or IP addresses to customize the password hash.

An advantage regarding the security of the hash value in the Catena framework is the ability to increase the parameter g to $g' = g + 1$ by calculating the new hash value h' with

$$h' = H(g'||(\lambda + 1) \cdot 2^{g'}||F(g', h)). \tag{5.33}$$

Due to the sequential structure, there is no need to involve the client, i.e., *client-independent update*. Due to the lambda λ value, the computation times can be increased by extending the depth of the graph approach, with values of 2 or 4 recommended. This approach can be used to update the hash values of the grid cells, thus protecting against brute-forcing of location information. This feature is provided to secure passwords even with increasing computational power without the necessary interaction of a service user, e.g., of a social network such as Twitter and Facebook. In GCH, this feature can be customized to increase the privacy of a target client's location information.

5.3.2 Probabilistic methods

The following section describes how a Bloom filter-based data structure is combined with the grid cell representation of geofences and the parent mechanism to enable a service-based privacy-preserving evaluation. Section 4.4.2 introduced the concept of Bloom filters, and Section 4.4.2.4 provided an overview of previous research in its adaptation of spatial evaluation.

5.3.2.1 Bloom filters

A Bloom filter is a space-efficient method to check whether an element is a set member. Therefore, it fits perfectly in using grid cells and their parent structure for privacy-preserving geofencing.

The represented values of geofence grid cells must be stored within the Bloom filter in the setup phase. In the service phase, it is necessary to check whether a parent of a target client's S2 point position is a member of the Bloom filter structure.

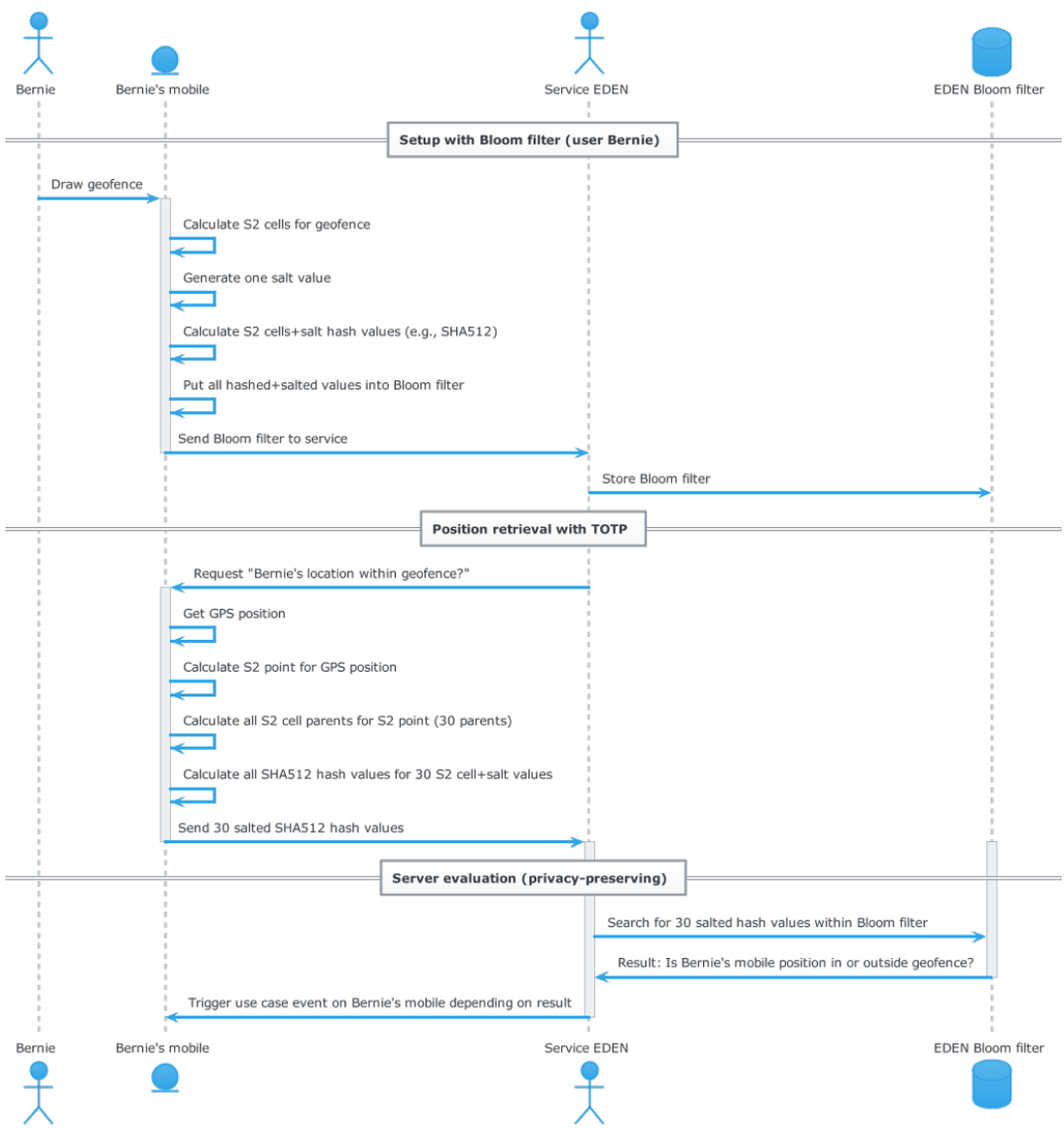


Figure 5.19: S2 grid cell hashing together with a simple Bloom filter for geofence evaluation through a service.

Although a Bloom filter uses hashing algorithms to store data, they are not intended to keep information secret. In the case presented, the client must still use a cryptographic hashing algorithm to hide the values of the grid cells from the service. This includes the use of a *secret*, i.e., *salt*, added to the grid cell value that only the client knows, see Figure 5.19.

The result of the setup phase is a Bloom filter set that the service stores along with the identity information of the particular user. After the setup, the client also sends hashed parent values to the service in the same way as in the GCH approach. Before the service checks the Bloom filter set, it must apply Bloom filter hashing to check the bucket values of the filter set.

5.3.2.2 Counting Bloom filters

The specific *Counting Bloom filter*, see Section 4.4.2, is used when the client wants to update the geofence values by sending hashed grid cell values to the service, where they are added to or deleted from the filter. These characteristics provide additional capabilities when a use case requires variable or changing geofence areas, such as the carsharing use case where on-the-fly congestions may be added or deleted from the geofence set. Also, the Bloom Filter's space-efficient way of storing set data is an advantage over storing grid cell hashes in traditional ways. Section 7.3.3 evaluates storing sizes of the Bloom filter approach.

Combining the grid cell hashing approach with any Bloom filter variant for privacy-preserving geofencing is referred to as *Probabilistic GCH* or *Bloom Filter GCH*. The main difference between this approach and the one by Bösch et al. [173] described in Section 4.4.2.4 is the lack of a trusted third party. Therefore, encryption and decryption are not used in the presented approach. Bösch's approach also does not support arbitrarily shaped geofences because it uses *coordinate values* for the geofence and a *use-case-dependent accuracy value*.

5.3.2.3 Inferred information and countermeasures

Since the parent values must be tested against the Bloom filter, the parent values *cannot* be masked with a *time-varying nonce* since this would interfere with the filter test. This fact leads to the situation where a service provider can derive information from a position's periodically received parent values. Since the salt value protects the parent values, the service may not know the target's position, but it can tell if the target is moving based on the stream of parent values. If the parent values change between position updates, the target moves, but if the values do not change, the target does not move.

This shortcoming can be overcome in different ways. Similar to the countermeasures discussed in the GCH section, e.g., the geofence could consist only of non-normalized values, and the values sent to the service could have a fixed level. The approach of using the inverted geofence for the evaluation is also viable. Another technique that takes advantage of the smaller memory sizes of Bloom filters is to populate the filter with values using multiple salt values. Thus, one can alter salt values at the client for its position updates. The last measure is suitable for use cases where the client needs a position update less frequently, such as the whereabouts use case.

5.3.3 Homomorphic encryption

This section addresses the extended variant of the NEXUS approach described in Section 4.4.3 using *homomorphic encryption* (HE). A combination of HE with zero-knowledge proofs is briefly described in Section 5.3.6.3. Relevant information on three specific HE schemes is summarized in Appendix A.2.

5.3.3.1 NEXUS parallel

As described previously, calculations on homomorphically encrypted ciphertexts are mostly limited to basic arithmetic operations. It has also been a privacy-enhancing mechanism to outsource complex computations by transferring only encrypted data to third parties, such as cloud providers.

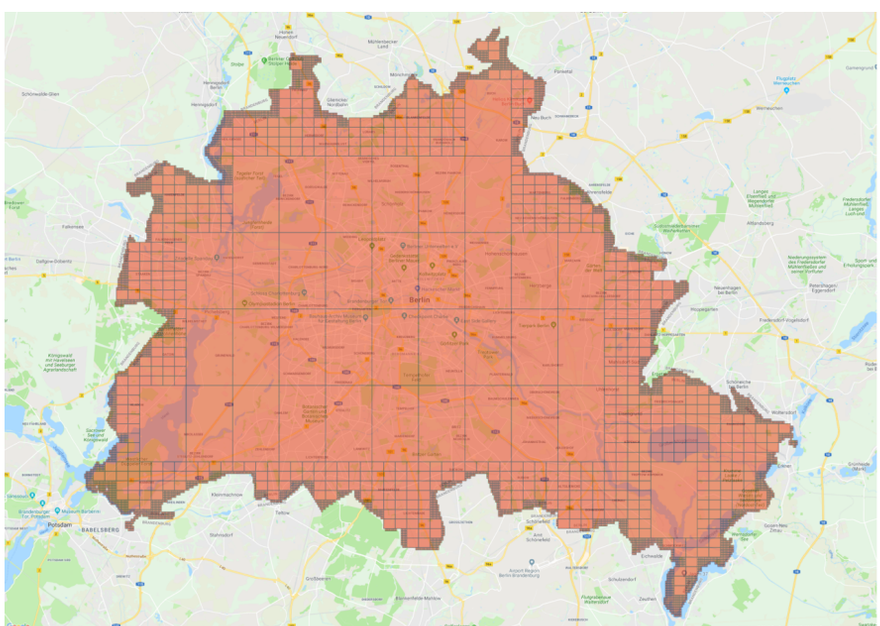


Figure 5.20: The geofence of Berlin represented by normalized Geohashes.[4]

This section describes how Geohashes can be used to extend the NEXUS approach [176] to be able to handle arbitrarily shaped geofences. This has already been mentioned in the outlook section of the original paper. The main drawback of the S2 cell structure is that the projected areas differ in their geometric shape depending on their Earth's location because a sphere is projected onto a cube. The Geohash geo encoding presented in Section 4.2.2 is convenient when applying an extension of the NEXUS approach, which builds its geofence representation rectangles with four long/lat coordinates as its corners. The evaluation uses perpendicular projection, dot products, and component-wise computation to assess whether or not a long/lat position lies within a rectangle or not. The main calculation is performed on homomorphically encrypted data.

Figure 5.20 shows a normalized geofence representation of Berlin using Geohashes. The NEXUS approach uses the mentioned vector arithmetic based on vectorized coordinates. The Geohash representation is well suited to represent a geofence. In the extended

[4]Image source: Underlying map data from Google Maps 2018 GeoBasis-DE/BKG (2009), Google. The figure was created using the QGIS open-source application.

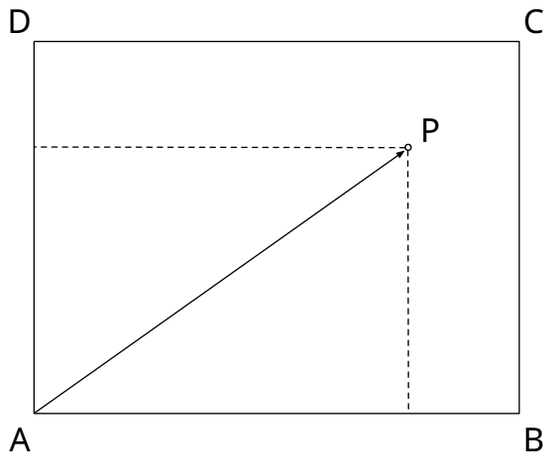


Figure 5.21: In the NEXUS approach, the figure shows the vector and perpendicular projection of position P with respect to a Geohash with vector points A, B, C, and D.

version of NEXUS, not only one rectangle (Figure 5.21) is evaluated against a long/lat position, but all normalized Geohash rectangles (Figure 5.22) capable of representing any shape. All Geohashes represent rectangles when projecting the normalized geofence polygon onto a 2D map. In the NEXUS approach, each Geohash is described by four corner points A, B, C, and D. The evaluation of the NEXUS variant is given in Section 7.3.4.

As mentioned in Appendix A.2.2, the Paillier encryption scheme supports *homomorphic addition* of ciphertexts, as seen in Equation A.8. It also supports the *homomorphic multiplication* of a constant and a plaintext, as seen in Equation A.9. The NEXUS approach uses these properties and the equations to *homomorphically subtract two plaintexts* by adding a negated ciphertext. The ciphertext can be negated by raising it to a negative value using Equation A.9. Based on the negation, one can determine whether value a is less than or equal to value b (after decryption) by checking

$$D_{\text{sk}}(E_{\text{pk}}(a-b)) \leq 0. \tag{5.34}$$

This has already been defined in Equation A.10 and is checked by the KA, which is enabled to only decrypt the result by using its secret key. In this HE approach, the TTP does not get any specifics about the actual position of the client. Each position P is a two-dimensional vector with long/lat values in the NEXUS implementation. To check if P is inside a rectangular geofence F = {A,B,C,D} as in Figure 5.21, Equation 5.35 is evaluated.

$$0 \leq \vec{\text{AP}} \cdot \vec{\text{AB}} \leq \vec{\text{AB}}^2 \wedge 0 \leq \vec{\text{AP}} \cdot \vec{\text{AD}} \leq \vec{\text{AD}}^2 \tag{5.35}$$

$E_{\text{pk}}(\vec{\text{AP}} \cdot \vec{\text{AB}})$ and $E_{\text{pk}}(\vec{\text{AP}} \cdot \vec{\text{AD}})$ must be calculated homomorphically, due to the encryption of position P.

For *longitude* and *latitude* values, the perpendicular projection of P can be calculated homomorphically as

$$E_{\text{pk}}(\vec{\text{AP}}_{\text{long,lat}}) = E_{\text{pk}}(\vec{\text{P}}_{\text{long,lat}}) \ominus E_{\text{pk}}(\vec{\text{A}}_{\text{long,lat}}) \tag{5.36}$$

and

$$E_{\text{pk}}(\vec{\text{AP}} \cdot \vec{\text{AB}}) = E_{\text{pk}}(\vec{\text{AP}}_{\text{long}})^{\vec{\text{AB}}_{\text{long}}} \oplus E_{\text{pk}}(\vec{\text{AP}}_{\text{lat}})^{\vec{\text{AB}}_{\text{lat}}}. \tag{5.37}$$

Similar to the longitude calculations in Equations 5.36 and 5.37, $E_{\text{pk}}(\vec{\text{AP}} \cdot \vec{\text{AD}})$ is calculated for the latitude values.

The result R in Equation 5.38 is sent encrypted to the KA. Depending on the decrypted check result, whether the position P is inside one of the rectangles or not, actions are triggered for the subscribed nodes.

$$\text{R} = \{\vec{\text{AB}}^2, \vec{\text{AD}}^2, E_{\text{pk}}(\vec{\text{AP}} \cdot \vec{\text{AB}}), E_{\text{pk}}(\vec{\text{AP}} \cdot \vec{\text{AD}})\} \tag{5.38}$$

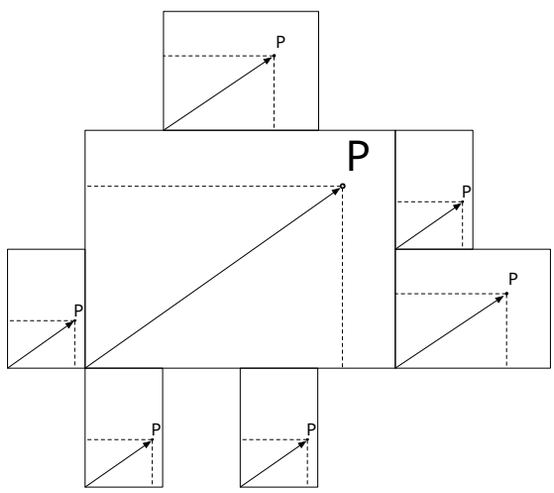


Figure 5.22: In the extended NEXUS approach, the vector and perpendicular projection of the target position P is evaluated using HE against all Geohashes of an arbitrarily shaped geofence. Each individual Geohash has four vertices (A, B, C, and D).

Since HE is suitable for scenarios where a trusted third party is present, the approach provides a fundament for Objective 16 (*services do not get shared secrets*).

5.3.3.2 Drawbacks of homomorphic encryption

The disadvantages of HE, not only in location-based privacy scenarios, are reasonably obvious. The entity holding the secret key can decrypt information. HE does not provide any benefits in the constellation where a TTP is not desired since the service would have the secret key. This would allow the service to decrypt an evaluation method indicating whether a target is in a geofence or not. However, this evaluation would be performed either on the client-side or by a TTP since the actual position should be hidden from the service. HE is intended for use in cloud computing use cases where the cloud service has high computational power, and the client-side encrypted data is homomorphically analyzed at the cloud service. After computing the encrypted data, the result is sent back to the client, where it can be decrypted and used. To enable such a cloud service, efficient, fully homomorphic encryption schemes need to be developed. Moreover, the HE approach described above does not follow the general set membership approach of the various other protocols. In terms of performance, HE was also evaluated along with the other techniques described in the dedicated Evaluation Section 7.3.4 and is therefore also included here as well.

5.3.4 Multi-party computation and PSI protocols

This section describes *multi-party computation* (MPC) protocols based on *public-key exchange*, *oblivious pseudorandom functions*, and *oblivious pseudorandom code*. In addition, *private set intersection* (PSI) protocols reveal only *membership cardinality* or even just *membership existentiality*. Chapter 4 introduced both main schemes, MPC in Section 4.4.4 and PSI in Section 4.4.5.

5.3.4.1 Diffie-Hellman-based

An early protocol introduced by Huberman et al. [222] and later classified among other PSI protocols by Pinkas et al. [223] is based on the Diffie-Hellman (DH) key exchange and the discrete logarithm problem. DH is defined in Appendix A.1.4. The public-key-based PSI protocol can be seen in the following equations. It is based on the associativity of the exponentiation operation. The client and service perform the following steps. Protocol 5.39 shows both participants and their computations during the setup and execution phases.

- Both agree on a cyclic group of prime order q.
- Both agree on a hash function $H()$.
- Both choose their individual secret α and $\beta \in_R \mathbb{Z}_q$ randomly.
- Both hash their set values x_n and y_m.
- Both raise their hashed-set values to the power of their respective secret.
- Both permute their intermediate results randomly Π.
- Both exchange the intermediate results.
- Both raise the received values to the power of their respective secret.
- The client sends its resulting set permuted to the service.
- The service can evaluate how many set elements are present in both initial sets by comparing both values of the result set. The service does not know which of the initial values match.

The advantages of this straightforward protocol are remarkable. Both parties can compute operations in parallel. It is easy to implement. The client can choose a new random α value for each round, e.g., to hide information about its movement. Due to the permutation of the exchanged values, it is difficult for the service to determine whether a grid cell in the geofence previously matched the client's position - only the cardinality of the matches is shown to the service. In general, this is 0 or 1, with the prerequisite that there is no overlapping geofence in set Y_{n_2}.

In the proposed protocol, the two parties do not know which elements are in the opponent's set. Since this is a privacy-preserving function, the service could use it to determine the client's location by explicitly identifying the elements in Y_{n_2}. The client can control this by letting it know the value β to enable geofence value checking. This can also be seen as a feature in the honest-but-curious security model, as there are use cases where the client should not know the geofence values or where the values change regularly.

The main disadvantage of this protocol is the communication costs. This includes the number of exchanged values ($n_1 + n_2$ and n_2) and two back-and-forth communications during execution. Likewise, the computational costs include $n_1 + 2n_2$ exponentiations and $n_1 + n_2$ hashing operations. During execution, the set X_{n_1} contains only 30 higher-order parent values. Furthermore, the values in the set Y_{n_2} can be hashed before execution.

The same applies to the exponentiation with β if the value is used unchanged for several successive evaluations. The hashing operation is not strictly necessary if the S2 integer representation is used for the grid cells. However, it is essential for sets with non-numeric elements. Both costs are less optimal than for the GCH approach. Pinkas et al. [223] note that communication overhead can be reduced using elliptic-curve cryptography, but this is considered future work.

Protocol using the
Diffie-Hellman key exchange

Client		**Service**
	Setup	
	Both agree on a cyclic group of prime order q and on a random oracle modeled hash function $H()$.	
chooses a secret $a \in_R \mathbb{Z}_q$		chooses a secret $\beta \in_R \mathbb{Z}_q$
	Execution	
computes set $X_{n_1} = \{(H(x_1))^a \ldots, (H(x_{n_1}))^a\}$		computes set $Y_{n_2} = \{(H(x_1))^\beta \ldots, (H(x_{n_2}))^\beta\}$
permutes order of results $\Pi(X_{n_1})$		permutes order of results $\Pi(Y_{n_2})$
	$\xrightarrow{\text{sends } \Pi(X_{n_1})}$	
	$\xleftarrow{\text{sends } \Pi(Y_{n_2})}$	
$\forall i \in \Pi(Y_{n_2}) = \{{i_1}^a, \ldots, {i_{n_2}}^a\} = \Pi(Y_{n_2}^a)$		$\forall j \in \Pi(X_{n_1}) = \{{j_1}^\beta, \ldots, {j_{n_1}}^\beta\} = \Pi(X_{n_1}^\beta)$
	$\xrightarrow{\text{send } \Pi(Y_{n_2}^a)}$	
		compares $\Pi(Y_{n_2}^a)$ with $\Pi(X_{n_1}^\beta)$
		if $x = y$ then $(H(x)^a)^\beta = (H(y)^\beta)^a$

(5.39)

5.3.4.2 Phasing

In 2015, Pinkas et al. presented the *Phasing* protocol (*Permutation-based Hashing Set Intersection*) based on the cryptographic primitives *cuckoo hashing* [170] and *oblivious pseudorandom functions* (OPRF). As mentioned before, cuckoo hashing is a probabilistic data structure (see Section 4.4.2.3), similar to the Bloom filter used in Section 5.3.2. OPRF is a two-party protocol that computes a joined pseudorandom function without revealing any secret values, see Appendix A.3.3. The protocol is also known as the Pinkas-Scheider-Segev-Zohner (PSSZ) construction [223].

The customized protocol, in which the roles are swapped, is used for privacy-preserving geofencing as follows, also shown in the client-service Equations 5.40.

- Andy has the role of the OPRF sender, i.e., the target.
- Becky has the role of the OPRF receiver, i.e., the LBS service.
- Both parties choose three hash functions $h_1(), h_2(), h_3()$ for 3-way Cuckoo hashing.
- Andy has inputs X and Becky has inputs Y, with $|X| = |Y| = n$.
- Becky maps her items into $1.2n$ bins using Cuckoo hashing and a stash of size s. She pads the remaining empty bins and the stash with dummy values to fill everything up.
- Both execute $1.2n + s$ instances of an OPRF function. Becky, the receiver, uses all items as input for the OPRF.
- $F(k_i, \cdot)$ denotes to the PRF evaluation in the i-th OPRF instance.
- Becky learns the output of Function $F(k_j, y)$ for the item y that was previously put into the bin j. when Becky has mapped the item y to the stash, she learns $F(k_{1.2n+j}, y)$.
- Andy can compute $F(k_i, \cdot)$ for each i. He computes two sets of possible PRF outputs

$$H = \{F(k_{h_i(x)}, x) | x \in X \text{ and } i \in \{1, 2, 3\}\}$$

$$S = \{F(k_{1.2n+j}, x) | x \in X \text{ and } j \in \{1, ..., s\}\}.$$

- Andy randomizes the order elements in the sets H and S and sends them to Becky.
- Becky can identify the intersection of the original sets X and Y by checking her OPRF outputs, i.e., by comparing the items y of the bins or the stash with the values in H and S.

In general, the parameters of the PRF must be set to avoid collisions. It should be emphasized that the result of the Phasing PSI protocol will provide the receiver with a set of values that include the intersection of the sets X and Y, i.e., $X \cap Y$. If the parent values of the S2 cell in both the senders and the receivers sets are not masked, the service (receiver) would know the position of the target client. How to mask grid cells was described in Section 5.3.1. Section 5.3.4.3 presents protocols that reveal only the cardinality and the existentiality of a set intersection.

Permutation-based Hashing Set Intersection (Phasing)

Client (Andy)		**Service (Becky)**
	Setup	
	Both parties agree on three hash functions $h_1(), h_2(), h_3()$.	
has set input $\|X\| = n$		has set input $\|Y\| = n$
		uses Cuckoo hashing, map Y
		to $1.2n$ bins and uses stash of size s
		pads all still empty bins and
		stashes positions with dummies
	Both run $1.2n + s$ instances of an OPRF function.	
	$F(k_i, \cdot)$ denotes to the PRF evaluation in the i-th OPRF instance.	
		output of $F(k_i, y)$ for item y if in bin i
		output of $F(k_{1.2n+j}, y)$ for item y if in stash j
	Protocol during execution	
computes $F(k_i, \cdot)$ for all i		
computes two possible PRF set outputs H and S		
$H = \{F(k_{h_i(x)}, x) \| x \in X \text{ and } i \in \{1, 2, 3\}\}$ and		
$S = \{F(k_{1.2n+j}, x) \| x \in X \text{ and } j \in \{1, \ldots, s\}\}$		
randomly permutes order of values in each set $\Pi(H)$ and $\Pi(S)$		
	$\xrightarrow{\text{send sets } \Pi(H) \text{ and } \Pi(S)}$	
		identifies $X \cap Y$, by checking OPRF outputs,
		i.e., items y of the bins or the stash
		against the values in H and S

(5.40)

Private Set Intersection Cardinality (PSI-CA) (all calculations are mod p)

Client	**Service**
Setup	
$C[j] = \{c_1, \ldots, c_w\}, \Pi(C)$	$S[i] = \{s_1, \ldots, s_v\}$
calculations for elements are done:	calculations for elements are done:
$\forall i 1 \leq i \leq w$	$\forall i 1 \leq i \leq v$
$p, q, g, k, H(), H'()$	
$\forall R : R_X \leftarrow \mathbb{Z}_q$ randomly	
	$R_S, R'_S, X = g^{R_S}$
	$h_{s_i} = H(s_i)$
	$a_i = (h_{s_i})^{R'_S}$
Initial transfer	
$\xleftarrow{X, \{a_1, \ldots, a_v\}}$	
$R_C, R'_C, Y = g^{R_C}$	
$a'_i = (a_i)^{R'_C}$	
$\Pi(a'_i)$	
$\xrightarrow{Y, \Pi\{a'_1, \ldots, a'_v\}}$	
Parents calculations	
$h_{c_j} = H(c_j)$	
$b_{c_j} = (h_{c_j})^{R'_C} \cdot X^{R_C}$	
$t_{c_j} = H'(b_{c_j})$	
Sending position update	
$\xrightarrow{t_{c_j} = \{t_{c_1}, \ldots, t_{c_w}\}}$	
Cardinality calculation	
	$b_{s_i} = (Y^{R_S}) \cdot (a'_i)^{1/R'_S \bmod q}$
	$t_{s_i} = H'(b_{s_j})$
	output $= \lvert\{t_{c_1}, \ldots, t_{c_w}\} \cap \{t_{s_1}, \ldots, t_{s_v}\}\rvert$

(5.41)

5.3.4.3 PSI variants

As defined in Section 4.4.5, a more restricted variant of PSI shows only the cardinality of the members the sets have in common (PSI-CA) [178]. This privacy-preserving method can be utilized to check whether a position is within a geofence. In the presented context, the cardinality is *one*. This is true for a geofence or a set of non-overlapping geofences. The general approaches of PSI-CA can be restricted even further since we only need to know if there *exists* a set member (PSI-X) [179].

PSI-CA with S2 In 2021 Cristofaro et al. published a private set membership cardinality protocol [178]. In its adaptation, the notion of the actors is changed to correspond to the functionality that the client sends position updates C to the server that holds the geofence information S, where the cardinality of the set intersection is evaluated. The result of this privacy-preserving protocol is the cardinality $|I|$ of the set intersection $I = C \cap S$. The whole process is divided into five steps.

In the setup phase, the calculations of the S2 geofence are performed. Secondly, the initial exchange of values is carried out. Thirdly, the calculation of the position and its masking is performed. Fourthly, the masked values are sent to the service. Finally, in the fifth step, the service calculates the cardinality of the set intersection. The detailed description of each step is as follows. This is also shown in the Equations 5.41.

1. **Setup phase**: The values of the S2 grid cells are calculated and stored in set $S = \{s_1, ..., s_v\}$. Two prime numbers, p and q, a generator g of a subgroup of size q, and two random oracle hash functions $H : \{0,1\} \rightarrow \mathbb{Z}^*{}_q$ and $H' : \{0,1\}^* \rightarrow \{0,1\}^k$, given a security parameter k are agreed upon. All R_X values are randomly chosen $\forall R : R_X \leftarrow \mathbb{Z}_q$. All grid cell values in S are being hashed with H. Each of the resulting values is raised to $(hs_i)^{R_S} = a_i$. Generator g is also raised to $R_S = X$.

2. **Initial transfer**: The values a_i and X are sent to the client. The client calculates $Y = g^{R_C}$ and raises each of the a_i values to the power of R'^C. The resulting values a'_i are randomly permuted and transmitted to the service, including the client value Y. X and Y are ElGamal public keys, as described in Appendix A.2.1.

3. **Parents calculations**: In this phase, the client calculates its position and all parents. These (current) parent values form the set C. The client permutes the order $\Pi(C)$ and hashes each value with H. The result will be calculated as $b_{c_j} = (h_{c_j})^{R'_C} \cdot X^{R_C}$ and hashed again with H'.

4. **Sending position update**: The resulting values are sent as set t_{c_j} to the service.

5. **Calculating the cardinality**: The evaluation of parents C against geofence S is done by applying the same cryptographic calculations to the values a'_i as were performed on the values just received. a'_i is raised to the power $1/R'_S \bmod q$ and multiplied by Y^{R_S}. The values b_{s_i} are then hashed with the same functions H' as values b_{c_j} on the client side in step 3 of the protocol. The cardinality output is calculated with $|\{t_{c_1}, ..., t_{c_w}\} \cap \{t_{s_1}, ..., t_{s_v}\}|$.

If $|\text{output}| = 1$, the target is inside the geofence, if $|\text{output}| = 0$ the target is outside. All calculations performed in this protocol are mod p. If there are overlapping geofences, the cardinality could be > 1. A privacy-enhancing mechanism should be considered. Regular changes to one or more R_X values should be implemented since the values do

not change for the client if the client does not move. The server would receive the same masked values. The levels are actually hidden because of the shuffling of values but may be extracted over time.

PSI existentiality There is the notion of PSI existentiality (PSI-X), which reveals a boolean value only if at least one element is present in both sets and nothing else. The protocol used here is an adaptation of the protocol by Carpent et al. [179]. Following the PSI-X introduction in Section 4.4.5, the prerequisites and the essentials for this protocol are as follows.

There are two sets, A for the client and B for the service. The two sets are represented by their characteristic functions $\chi_A()$ and $\chi_B()$. The individual elements are compared at position i represented by the product $\chi_A(i)\chi_B(i)$ and their aggregation is the disjunction

$$\bigvee_{\forall i} \chi_A(i)\chi_B(i) \text{ which is 1 iff } A \cap B \neq \emptyset. \tag{5.42}$$

Equation 5.42 can be evaluated on encrypted data by using the partially homomorphic encryption scheme *Boneh-Goh-Nissam* (BGN) [224], see Appendix A.2.3 for details. This scheme allows the evaluation of encrypted 2-DNF formulas (disjunctive normal form), also known as *OR of ANDs* or *sum of products*. This scheme allows the homomorphic operations *multiple-once* and *add-many*.

Equation 5.43 shows the number of unique S2 values, i.e., the size of the universe of set elements U, see also Section 4.2.

$$\sum_{n=0}^{30} 6 * 4^n = 9223372036854776000 \approx 9 * 10^{18} = |U| \tag{5.43}$$

Since U is very large, instead of its characteristic function, a 2-universal hash function $h() : U \rightarrow [1, ..., N]$ is randomly chosen from a family of hash functions H to avoid false positives. Equation 5.42 checks whether $h(A) \cap h(B) \neq \emptyset$ by evaluating $h(A)$ and $h(B)$.

The bit-vectors, UA for A, are calculated as shown in Equation 5.44.

$$\text{UA}_i = 1 \Leftrightarrow \exists x \in \tilde{A} : h(x) = i \tag{5.44}$$

The following page shows the protocol, also adapted from [179]. As with the PSI-CA protocol, the left and right sides are modified to use it in the presented scenario, where the service checks if $A \cap B \neq \emptyset$ or not.

A detailed protocol description of the Equations 5.45 is as follows.

- The client user and the service agree on a set of geofence values. These values are not private.
- The service stores the values of the geofences as set B.
- The client calculates its position and generates the parent values as set A.
- The set A of the client is blinded using a *keyed pseudorandom function* (PRF) using k_{prf}.
- The set B is blinded using an oblivious evaluation of the PRF (for OPRF, see Appendix A.3.3) using the same key k_{prf}. The oblivious evaluation enables forward security.

- The service initiates a homomorphic encryption key pair according to Boneh-Goh-Nissim and sends the public key k_{pub} to the client [224].
- The following steps are repeated R times:
  - A 2-universal hash function is randomly chosen by the two parties from a family of hash functions H.
  - Two-bit vectors are built UA and UB.
  - The bit vectors are encrypted by both parties as EUA and EUB, respectively.
  - This process is bundled so that only one exchange is done after the loop.
- Loop ends.
- The service sends its EUB_r values to the client.
- The client selects s to randomize the intermediate result X_r using homomorphic properties of the BGN scheme and returns it to the service.
- The service tests whether the encrypted value is 0 or 1. If $A \cap B \neq 0$, the output is always 1. If $A \cap B = 0$, the output is 0, with probability $1 - P^R$.

Private Set Intersection Existentiality (PSI-X)

Client		**Service**
	Setup	
sets geofence values B		
	$\xrightarrow{B}$	
		$B = \{b_1, \ldots, b_m\}$
	$k_{\text{prf}} \leftarrow_{random} \{0,1\}^l$	
	Protocol during execution	
$A = \{a_1, \ldots, a_n\}$		
$\tilde{A} \leftarrow \text{PRF}_{k_{\text{prf}}}(A)$	$\xrightarrow{\tilde{B}=\text{OPRF}_{k_{\text{prf}}}(B)}$	
	$\xleftarrow{k_{\text{pub}}}$	$(k_{\text{priv}}, k_{\text{pub}}) \leftarrow \text{BGN-Init}()$
	Loop for $r = 1..R$	
$\text{UA}_r \leftarrow (0, \ldots, 0), \lvert\text{UA}_r\rvert = N$		$\text{UB}_r \leftarrow (0,\ldots,0), \lvert\text{UB}_r\rvert = N$
for $i = 1..n$: $\text{UA}_{r,h_r(\tilde{A_i})} \leftarrow 1$		for $i = 1..m$: $\text{UB}_{r,h_r(\tilde{B_i})} \leftarrow 1$
$\text{EUA}_r \leftarrow \text{BGN-Enc}_{k_{\text{pub}}}(\text{UA}_r)$		$\text{EUB}_r \leftarrow \text{BGN-Enc}_{k_{\text{pub}}}(\text{UB}_r)$
	End loop	
for $r = 1..R$:	$\xleftarrow{(\text{EUB}_1, \ldots, \text{EUB}_R)}$	
$s \leftarrow_{random} \{0,1\}^{l'}$		
$X_r = s(\text{EUA}_r \cdot \text{EUB}_r)$		
	$\xrightarrow{(X_1, \ldots, X_R)}$	if $\exists r$: $\text{BGN-Is-Zero}_{k_{\text{priv}}}(X_r)$:
		return 0
		else
		return 1

(5.45)

The presented MPC and PSI protocols differ in the information that is revealed to the involved parties. The PSI-X protocol is the one that does not disclose any additional information other than the fact that there is or is not at least one element present in the sets. The evaluation chapter will give an overview of how the objectives are being addressed by the general MPC protocols and the specific PSI protocols, see Section 7.2.

5.3.5 Cryptographic accumulators

Benaloh and de Mare introduced the term *accumulators* starting with one-way accumulators [180]. Along with this early notion, other accumulator schemes were developed,

as described in Section 4.4.6. The reasons for using this data structure within the P3IP approaches are manifold.

- It is a space-efficient way to store and analyze data.
- CAs are ideal for membership testing. This is already mentioned in the original paper [180].
- There is no need for a trusted third party.
- Target users can add S2 cells to the accumulator.
- CAs are advanced over similar data structures, such as Bloom filters.

The following sections describe the initial use case of checking if a target client's position is within a set of values in general and will also focus on the use case where the target client is a carsharing vehicle. Accumulators will be used to verify whether or not the car was within a congested traffic area – in a privacy-preserving manner.

5.3.5.1 Elementary accumulator schemes

First, an elementary accumulator scheme is used, which corresponds to the overall approach of testing set membership. The basic methods $Gen()$, $Eval()$, $Wit()$, and $Ver()$ are used.

- $Gen()$, generates the key k.
- $Eval()$, uses k and inputs the gf as $\leq s2c_1 - s2c_3$, outputs the accumulator z together with aux (the geofence cells need to be hashed, see witness, Section 4.4.6)
- $Wit()$, actually, the witness function already returns the result if a value is within the set L if it outputs a w and not if the output is $\perp$.
- $Ver()$, returns *yes* or *no* on input (k, y_i, w_i, z). The problem is that for each y_i, you need a new witness w_i.

The time stamping example in Fazio et al. [181] is modified and adapted for the traffic congestion use case. The target client wants to prove, perhaps even later, that it was inside a traffic congestion geofence during a sharing process. This is somewhat like proving that a document has been time-stamped.

At the time t_x, the service generates an accumulator key k_x. The service then encodes the areas $y_{t,i}$ with traffic congestions from the overall domain of Y_{k_t}, i, e.g., with (hashed) S2 grid cells. All these areas, i.e., the $y_{t,i}$ values, for the time t_1 are accumulated with the output (z_t, aux_t) using the $Eval()$ method, which is described in Section 4.4.6.2.

The target client in the car receives either the values k_t and aux_t and uses its position y_t at the time t_i to generate a witness $w_{t,i}$ for it. The target client can then prove that it was at a particular position at the given time because of the unique values of k and the corresponding witness that proves that the position is a single value in the accumulated geofence values of all traffic congestion within a city or country. The number of congestions is related to the level of privacy, i.e., the more congestions areas are used, the higher the level of privacy for a specific user.

To prove that the target client was inside a traffic congestion, the target client sends all encoded position y_{t,i_x} and the witnesses w_{t,i_x} to the service. The service uses the $Ver()$ method together with its values k_{t_x} and z_{t_x} and the client's input to verify if there is a match in one or more of the timeframes. The position should not be revealed to the service, so encoding using previously described techniques, such as GCH, is required.

5.3.5.2 Dynamic accumulator schemes

By switching the roles, the client accumulates its position using the $Add()$ method of a dynamic accumulator scheme and then sends the result to the service frequently. The service checks if one or more cell values of the congestion areas have been accumulated. The dynamic accumulator schemes allow adding and deleting values to and from the accumulated value z. This mechanism is used to enable the dynamic geofence use case.

The execution roles described in the previous subsection are changed. In this approach, the target client will add new locations y_{t_x} to the accumulator on a regular interval z_x at given time intervals t_x, also using different keys k_{t_x}. This is necessary to ensure that there is no overlap between the client's positions and areas where there is no traffic congestion for the service to query. To prevent the service from querying every position at every time, different time-dependent k_{t_x} values are used.

For the service not to be able to test any location at any time and find out where the car was, the setup either relies on the service to use only relevant geofence data (semi-honest) or there is an official congestion oracle that provides reliable data.

If the roles change, the service can test any y_i, i.e., any area, and pass this together with the value k_{t_x}, w_i, z to the verification method. Thus the service can find out any client location, even outside of congestions. The service would have to rely on the use of only y_i values and their corresponding time-based keys k_{t_x}.

From a privacy-preserving point, the question is: Can the *aux* value be used to ensure that the client only produces witnesses for areas at a given time? However, the problem may be that the service uses all non-congestion values to check where a client has been with the $Ver()$ method. The client is asked if it was at point x at time t. The client proves that it was by presenting the witness w for the position x based on the time-dependent k. This is a proof-of-location use case for evaluating dynamic geofences, also for a post-service consumption evaluation. Perhaps the service receives official congestion reports for an entire day from a third party. The service uses this information and requests the client to prove its location to reduce the carsharing cost due to a traffic situation. In general, UAs are more used as a proofing mechanism rather than a privacy-preserving evaluation.

5.3.5.3 Universal accumulator schemes

As mentioned in Section 4.4.6, Benaloh and de Mare [180] introduced an accumulator based on the RSA groups. RSA is discussed in Appendix A.1.5. This *RSA accumulator* can handle sets of prime numbers. To use this approach, an appropriately constructed hash function is needed that maps arbitrary strings to primes since prime factorization is at the heart of the accumulated value A. The public parameters are as follows.

- Two large prime numbers p, q.
- They form the RSA modulus $n = p * q$.
- A random generator $G \in_R \mathbb{Z}_N^*$. $\mathbb{Z}_N$ denotes to the ring of non-negative integers modulo $n = \{0, 1, 2, ..., N-1\}$. $\mathbb{Z}_N^*$ is a subset of elements in $\mathbb{Z}_N$ which are coprime with N, i.e., they form a group under multiplication.
- A collision resistant hash function $H_{prm}()$ that maps an input x to a *prime* number e, $H_{prm}(x_i) = e_i$.
- The members of set $S = \{x_1, ..., x_n\}$, each converted to a prime number $S' = \{e_1, ..., e_n\}$. Again this assumes that S is very large; otherwise, it could

be uncovered with brute force. To prevent this, a large random prime number can be used into $\prod_{i=1}^{n} e_i$, which acts as a *salt* value.

The protocol is as follows (calculations are mod n). It is also reflected in the client-service Equations 5.53.

- Using the prime numbers in S' in the exponentiation product, a unique accumulator A of the set elements e_i is calculated as

$$A \leftarrow G^{\prod_{i=1}^{n} e_i}. \tag{5.46}$$

- The proof π for a set element $e_j \in S'$ is computed as

$$\pi_j \leftarrow G^{\prod_{i=1, i\neq j}^{n} e_i} = A^{\frac{1}{e_j}}. \tag{5.47}$$

- The proof π_j can be verified with

$$\pi_j^{e_j} \stackrel{?}{=} A. \tag{5.48}$$

In addition to the simplicity and elegance of this approach, it also has the advantages of having constant sizes for the accumulator and the proof values, and it only requires one modular exponentiation for the verification. This is a universal accumulator scheme since values can be added dynamically, non-membership, and many element proofs can be created.

Set membership in zero-knowledge In the CA protocol above, the value e_j is known to the verifier. The protocol needs to be extended to make it privacy-preserving, as the verifier should not know the client's parent values accumulated in A.

To achieve this, a zero-knowledge proof is constructed that proofs e_j is an element in A and that a property $P()$ holds for e_j in S' without revealing it. A ZKP for the NP relation $R(S', e_j) := x \in S' \wedge P(e_j)$ where S' is public and e_j is secret must be found. More precisely, a ZKP containing an RSA-based cryptographic accumulator must be constructed for the relation

$$R(A, (e_j, \pi_{e_j})) := \mathit{Verify}_{\text{mem}}(A, e_j, \pi_{e_j}) \wedge P(e_j) \tag{5.49}$$

in a manner that preserves privacy.

Using the Pedersen commitment scheme [90] described in Appendix A.1.8.1, Camenisch and Lysyanskaya [183] designed a ZKP protocol for knowing an integer e in A which is committed to a group $\mathbb{G}_q$. In detail, the public values are the accumulator A and a Pedersen commitment $C_e \in \mathbb{G}$. Camenisch and Lysyanskaya show how to prove the knowledge of (e, π, r) so that

$$A = \pi^e \wedge C_e = g^e h^r \in \mathbb{G}. \tag{5.50}$$

π is the *witness* of the RSA-based example. It is used as seen above with the *public* accumulator value as $A^{\frac{1}{e}}$ since the client knows e.

In the geofencing use cases, a service may publish its accumulator value(s) as public knowledge and make them available to the client. If the service were to make the prime values of A also public, the client could simply evaluate them using A. The technical disadvantage of using *prime numbers* as set elements does not make this otherwise elegant method straightforward.

5.3.5.4 Dynamic universal schemes

Since the described RSA-based CA is universal and can provide non-membership proofs, inverse geofences offer themselves to this task. In 2007, Li et al. [225] introduced this efficient universal accumulator to construct non-membership proofs. The term *universal* was chosen because, for every value in a domain, there is a witness for either membership or non-membership. The authors also proposed the notion of a *dynamic universal accumulator* that allows for the additions and deletions of values to and from the accumulated set A, and updates made to the witness values π. The article also describes the usage of a ZKP using commitments based on the related work by Camenisch and Lysyanskaya [183].

The non-membership verification for the RSA-based accumulator described above, as defined by Li et al. [225], is as follows

$$Verify_{\text{non-mem}}(A, e_n, a, d, g) = 1 \tag{5.51}$$

if and only if

$$A^a = d^{e_n} g \ (mod\ N). \tag{5.52}$$

In this verification, (a, d, g) combined is the non-membership witness, where $g \in QR_N : g \neq 1$ is a public value. The function $f_N(g, x) = g^x \bmod N$ and the accumulators' auxiliary information aux_f for f_N is the factorization of N are also needed for the generation of non-membership witnesses. The geofence can be represented as its inverse counterpart, and the client provides a non-membership proof for its S2 parent cell values to prove that it is within a private area. To create a non-membership proof, a set N contains some elements that do not belong to the committed set S and those that do, represented by the subset R of S. The *greatest common divisor* (gcd) of N and S is R. The client uses the extended Euclidean algorithm to find the coefficients (a, d) above, allowing the service to verify this fact. See [225] for details on computing non-membership witnesses with and without a trusted third party.

5.3.5.5 Merkle Tree accumulators

There are other approaches to efficient set membership testing in zero-knowledge using accumulators based on the data structure introduced by Ralph Merkle in 1987 [226]. These so-called Merkle Trees and accumulators have recently been revived in the context of distributed ledger technologies and blockchains. As Merkle Trees and ZKPs are computationally more demanding than RSA-based CAs, this is considered future research. In a sense, accumulators can be described as the asymmetric cryptography cousin of Merkle Trees and Bloom filters.

RSA-based Cryptographic Accumulator

Client	**Service**
Setup (standard geofence)	
Agree on two random primes p and q	
with RSA modulus $N = p \cdot q$ and generator $G \in_R \mathbb{Z}_N^*$	
plus random values g and h from group $\mathbb{G}_q$	
Check properties of p, q, g, h	
Agree on public values $H_{\text{prm}}()$	
	Set members of geofence $S = \{x_1, \ldots, x_n\}$
	Create set of primes $S' = \{H_{\text{prm}}(x_1), \ldots, H_{\text{prm}}(x_n)\} = \{e_1, \ldots, e_n\}$
	and accumulate them $A = G^{\prod_{i=1}^{n} e_i}$ (5.53)
$\xleftarrow{A}$	
Execution (without ZKP)	
Calculate position and its parent values $P = \{p_1, \ldots, p_{30}\}$	
Calculate primes $P' = \{H_{\text{prm}}(p_1), \ldots, H_{\text{prm}}(p_{30})\} = \{e_1, \ldots, e_{30}\}$	
Calculate witnesses $\pi_x = A^{\frac{1}{e_x}}$	
To commit to values from P' choose randomly r_x for each e_x	
Calculate commit values $C_{e_x} = g^{e_x} \cdot h^{r_x}$	
Send these sets for membership tests	
$\xrightarrow{\pi_x, e_x, C_{e_x}, r_x}$	
	$\text{Verify}_{\text{mem}}(A, e_x, \pi_x)$
	with $A = \pi_x^{e_x} \wedge C_{e_x} = g^{e_x} h^{r_x} \in \mathbb{G}$

5.3.6 Zero-knowledge proofs

Using the concept of *zero-knowledge proofs* of knowledge in a *non-interactive* way, the target must commit to the grid cell parent values of the current location. The created proof indicates whether the commitment is within the set of geofence values for each location update. An introduction to ZKPs was given in Section 4.4.7.

5.3.6.1 Anonymized RSA-based accumulators

In the previous section, the notion of cryptographic accumulators was used to verify a set membership. The presented protocols need to be extended for a privacy-preserving data evaluation. As mentioned earlier, Camenisch and Lysyanskaya [183] extended the RSA-based CA protocol by using zero-knowledge approaches. Li et al. [225] also extended the CA protocol with anonymity. The difference is that the latter describes an anonymous non-membership proof for its universal accumulator, and the former describes a ZK membership proof.

This difference lends itself to the use of inverse or regular geofences. In an inverse geofence setup based on Li et al. [225], the client proves – in zero-knowledge – that it can generate a non-membership proof for its current location. If the client fails to do so, it is outside the privacy region, i.e., inside the inverse geofence. Similarly, given a geofence setup and the previously mentioned Camenisch and Lysyanskaya ZKP protocol [183], the client generates a zero-knowledge membership proof for the privacy area.

In CA's wording, this means generating a proof that shows in ZK that a committed value hash has been accumulated or not, i.e., it is in the geofence set or not. The two approaches are based on commitment schemes, such as the Pedersen scheme, see Appendix A.1.8.1.

In [225], the non-membership proof of knowledge (PK) is as in Equation 5.54.

$$\mathrm{PK}\{(x, r, a, d) : c_1 = g_1^x h_1^r \wedge c^a = d^x g \wedge x \in \mathbb{Z}_{2^l} \wedge a \in \mathbb{Z}_{2^{l'}}\} \tag{5.54}$$

Li et al. [225] prove the following theorem for this protocol.

The protocol is a zero-knowledge proof of knowledge of the values (x, r, a, d), such that $c_1 = g_1^x h_1^r \bmod n_1$ and (a, d) is a valid non-membership witness of x for accumulator c.

5.3.6.2 Zero-knowledge set membership algorithm

The following protocol is an adaption from [227], a zero-knowledge proof construction for set membership queries (ZKSM). The protocol is based on cryptographic primitives that are described in Chapter 3 and Appendix A.4.1.

The idea behind the protocol is as follows. For each element $i \in S$, the verifier signs the value using a random value x and the RSA digital signature scheme (Appendix A.1.5.1, steps 1-3). The prover blinds the values A_i by raising each signature value to a random exponent τ, resulting in the set V (Step 4).

For Step 5, recall the concept of proofs of knowledge (PK) from Section 4.4.7.1 and Equation 5.55.

1. Input parameters: $g, h,$ a commitment C, and a Set S.
2. Prover input: δ, γ such that $C = g^\delta h^\gamma$ and $\delta \in S$.

3. Verifier picks: $x \in_R \mathbb{Z}_p$. and sends: $y \leftarrow g^x$ and $A_i \leftarrow g^{\frac{1}{x+i}}$ for every $i \in S$.
4. Prover picks: $\tau \in_R \mathbb{Z}_p$ and sends $V \leftarrow A_\delta^\tau$.
5. Prover and verifier run: PK $\{(\delta, \gamma, \tau) : C = g^\delta h^\gamma \wedge V = g^{\frac{\tau}{x+\delta}}\}$.
6. Prover picks: $s, t, m \in_R \mathbb{Z}_p$. and sends: $a \leftarrow e(V,g)^{-s}.e(g,g)^t$ and $D \leftarrow g^s h^m$
7. Verifier sends random challenge: $c \in_R \mathbb{Z}_p$.
8. Prover sends: $z_\delta = s - \delta c, z_\tau \leftarrow t - \tau c, z_\gamma \leftarrow m - \gamma c$.
9. Verifier checks: $D = C^c h^{z_\gamma} g^{z_\delta}$ and $a = e(V,y)^c.e(V,g)^{-z_\delta}.e(g,g)^{z_\tau}$.

A computational evaluation of this approach based on the referenced library can be found in Section 7.3.5. In the following proof of implementation chapter, Section 6.1.4.3 shows a program code example based on the presented algorithm.

Zero-knowledge proof for set membership testing

Client (prover)		**Service (verifier)**
	Setup	
	$g, h,$ a commitment C, and a Set S	
δ, γ such that		
$C = g^\delta h^\gamma$ and $\delta \in S$		
		$x \in_R \mathbb{Z}_p$
		$y \leftarrow g^x$ and $A_i \leftarrow g^{\frac{1}{x+i}}$ for every $i \in S$
	$\xleftarrow{y, A_i}$	
$\tau \in_R \mathbb{Z}_p$		
$V \leftarrow A_\delta^\tau$		
	$\xrightarrow{V}$	
	Both run	
	PK $\{(\delta, \gamma, \tau) : C = g^\delta h^\gamma \wedge V = g^{\frac{\tau}{x+\delta}}\}$	
	During execution	
$s, t, m \in_R \mathbb{Z}_p$		
$a \leftarrow e(V,g)^{-s}.e(g,g)^t$		
$D \leftarrow g^s h^m$		
	$\xrightarrow{a, D}$	
		random challenge$c \in_R \mathbb{Z}_p$
	$\xleftarrow{c}$	
$z_\delta = s - \delta c$		
$z_\tau \leftarrow t - \tau c$		
$z_\gamma \leftarrow m - \delta c$		
	$\xrightarrow{z_\delta, z_\tau, z_\gamma}$	
		check
		$D = C^c h^{z_\gamma} g^{z_\delta}$ and
		$a = e(V,y)^c.e(V,g)^{-z_\delta}.e(g,g)^{z_\tau}$

(5.55)

5.3.6.3 XOR and homomorphic multiplication

To evaluate whether all 64 bits of two S2 cells are the same, one approach is to use the binary operation of a bitwise XOR ($\oplus$). It takes two-bit patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The

Table 5.4: XOR result of two non-equal S2 cells

	0100	0111	1010	1000	0101	0001	1010	1100	
	0100	0111	1010	1000	0101	0001	1011	0000	
$\oplus$	0000	0000	0000	0000	0000	0000	0001	1100	$\neq 0$

Table 5.5: XOR result of two equal S2 cells

	0100	0111	1010	1000	0101	0001	1010	1100	
	0100	0111	1010	1000	0101	0001	1010	1100	
$\oplus$	0000	0000	0000	0000	0000	0000	0000	0000	$= 0$

result is 0 if both bits are equal, i.e., if they are both 0 or 1 [228].

Tables 5.4 and 5.5 show two examples. The S2 cells are not equal in the first table, which means that the resulting $\oplus$ produces a non-zero result. In the second table, the S2 cells are equal, which results in a zero, 0.

That means that for a position evaluation against the geofence, all 30 S2 parent values $|n|$ for an S2 point need to be checked (XORed, $\oplus$) against all geofence grid cells $|m|$. Of course, this needs to be done in a zero-knowledge fashion since the service does not receive any plaintext parent value.

To calculate the final result $|B_{\oplus}| = |n_{parents} \cdot m_{gf-cells}|$, a multiplication of all $\oplus$ results using homomorphic encryption is executed, see Equation 5.57. If one parent value matches any of the geofence cells, the result of the $\oplus$ operation is zero, as explained above. The product of all $\oplus$ results in $b_{\oplus_i}$ to a single value yields either results in zero or in an arbitrary binary number. If the result is zero, the client is inside the geofence. If the result is not zero, the client is outside the geofence, see Equation 5.56.

$$\prod_{i=1}^{n \cdot m} b_{\oplus_i} \stackrel{?}{=} 0 \tag{5.56}$$

A homomorphic encryption scheme to support ciphertext multiplication with respect to its plaintext multiplication is the ElGamal scheme introduced earlier and described in detail in Appendix A.2.1.

$$E_{\mathrm{ElG}}(b_{\oplus_0}) \cdot E_{\mathrm{ElG}}(b_{\oplus_1}) \cdot \ldots \cdot E_{\mathrm{ElG}}(b_{\oplus_{n \cdot m}}) \stackrel{?}{=} E_{\mathrm{ElG}}(0) \tag{5.57}$$

A service holding the private key of such a scheme should be able to decrypt and respond to the result of a client's ciphertext multiplication.

5.3.7 Attribute-based Encryption

The ABE definitions were given in Section 4.4.8. The term Key-Policy ABE (KP-ABE) notion was briefly mentioned, along with an introduction to Ciphertext-Policy ABE (CP-ABE). In the following subsection, KP-ABE is used in conjunction with inverted geofences to provide precise position disclosure for targets that are not within a pre-defined geofence. This requirement is motivated by the doping control use case and generalized in Objective 8.

5.3.7.1 Inverse geofences with position disclosure

In the doping control use case, a requirement stated that an athlete's position is only disclosed to the service if it is outside of a self-defined geofence, i.e., outside of a privacy garden. This scenario can be realized by including a policy within the *secret key* and an *inverted geofences* system. That is the reason why KP-ABE is utilized. The process can be summarized as follows.

1. The athlete defines her privacy geofence, e.g., an area within a city.
2. The client calculates the inverted geofence.
3. The athlete creates the keys during setup, i.e., a public key and a secret master key.
4. The athlete creates a secret key, i.e., a private key, containing the inverted geofence as an access policy.
5. The athlete sends this secret key to the service.
6. The service requests the athlete's position during the execution phase.
7. The athlete's device calculates the current position and encrypts the message to the service, including the S2 grid parent values of its geo encoded position as ABE attributes. Her exact long/lat coordinates m are included in the encrypted message $E(m)$.
8. This encrypted message is sent to the service.
9. The service attempts to decrypt the message.

   9.1 If the ABE attributes do not match the policy in the private key, the message cannot be decrypted, and the message m, i.e., the long/lat position, is not revealed, which also means that the athlete is within one of her privacy geofences. The DCO may receive information about an obfuscated position, see Section 5.3.9.

   9.2 If the message m can be decrypted with the athlete's generated private key, the policy in the key matches. This means that the athlete is not within a privacy geofence, as she is in an inverted one. Therefore, the exact long/lat position m is accessible to the service, i.e., the message is decrypted. The DCO can use the position to carry out a doping control.

Inverted geofences, introduced in Section 5.2.2, are exemplified by their usage in the ABE scheme. They can be used with other approaches to hide user positions as an additional privacy layer.

5.3.7.2 Key policy ABE

In this section, the previous approach is described formally. The term *skey* for the secret key is used to distinguish it from the secret master key *smasterkey*

The client C generates a *public key* (*pubkey*) and a *secret master key* (*smkey*).

$$generate() : pubkey_C, smkey_C \tag{5.58}$$

The target generates a *secret key* (*skey*) that contains the values of the encoded inverse geofence using either the logical *or* or *of* operator in its *key-policy* (KP), see examples

below.

$$generate(smkey_C) : skey_S \text{ with policy } gf_{inv} \tag{5.59}$$

The target sends this *skey* to the service.

When the client's position is requested, it encrypts $E()$ its exact position as the message m_C together with all position *parent* values $parents_C$ as ABE attributes using the *pubkey*.

$$E_C((m = pos_C), parents_C, pubkey_C) \tag{5.60}$$

The service receives the *encrypted message* and uses its corresponding *skey* to decrypt the *message*. If any of the *attributes*, i.e., the values of the client's *parents*, match any *cell* in the policy of the *skey*, the service *can* decrypt the message and gets to know the client's position. This is possible only if the target is outside its defined geofence, i.e., inside the inverse geofence. If the message *cannot* be decrypted, the target is within one of its pre-defined geofences, and the location privacy is preserved.

$$|parents_C \cap gf_{inv}| > 0 \Rightarrow D_{skey_s}(E(m)) = pos_C \tag{5.61}$$

The *message* may contain additional information, such as a user or target id or device or service information.

In this approach, key handling is done on the target side. The target can generate new *secret keys* with *different policies*, i.e., *change* or *add* geofences. The control of generating keys lies with the target.

Since the current ABE libraries support strings with a leading character, the encoded geofence policy can be defined as follows (S2 and Geohash examples are for the Berlin area).

- Geohash strings: (gh_u337 *or* gh_u33e *or* gh_u334 *or* gh_u336 *or* gh_u33f *or* gh_u333 *or* gh_u339)
- S2 integers: (s2int_5163416555198873600 *or* s2int_5163434147384918016 *or* s2int_5163442943477940224 *or* s2int_5163464933710495744)
- S2 hex tokens: (s2hex_47a824 *or* s2hex_47a834 *or* s2hex_47a83c *or* s2hex_47a85 *or* s2hex_47a864 *or* s2hex_47a86c)

Also, the *of* operator can be used within the ABE policy.

- 1 *of* (s2hex_47a824, s2hex_47a834, s2hex_47a83c, s2hex_47a85, s2hex_47a864, s2hex_47a86c)

Key-Policy-ABE

Client		**Service**
	Setup	
generate $pubkey_C$ and $smasterkey_C$		
calculate and encode inverse geofence gf_{inv}		
generate $skey_S$ with policy gf_{inv} from $smkey_C$		
	$\xrightarrow{skey_S}$	
		store $skey_S$ with uid_C
	Execution	
	$\xleftarrow{\text{request position of client}}$	
calculate current client position pos_C and $parents_C$		
encrypt $E_C((m = pos_C, parents_C), pubkey_C) = E_C(m)$		
	$\xrightarrow{E_C(m)}$	
		try to decrypt, if $\lvert parents_C \cap gf_{inv} \rvert > 0$ $\Rightarrow D_{skey_S}(E_C(m)) = pos_C$

(5.62)

5.3.7.3 Inferred information and countermeasures

It should be emphasized that without blinded cell values, an ABE scheme must be chosen that does not disclose the attributes sent with the encryption. If this is the case, the service could examine the attributes. If a masking mechanism is used to prevent this, it is necessary to use additional means to protect the initial set of geofence cells within the KP access policy. It is also required that each target client generates its own secret master key to prevent the service from decrypting the positions of other targets.

By inserting the geofence attributes into the private key and the geofence policy into the ciphertext, i.e., concatenating the parent values with the *or* operator, one could also utilize CP-ABE for privacy-preserving geofencing since one of the attributes would match the access policy within the ciphertext.

If the ABE implementation supports the *delegation* function, it allows the service to create private keys with a subset of attributes. The service can create private keys with single attributes that represent only a smaller encoded area, i.e., a grid cell. If each of these keys is used, the service can narrow down the target position to a single encoded area, regardless of the geo encoding system. To avoid this behavior, an ABE implementation without delegation functionality is needed. However, with inverted geofences, the delegation function would only reveal areas outside the defined privacy geofences where the presented protocol reveals the target position.

5.3.8 Private pattern matching

The two variants of *secure pattern matching* (SPM) described in Section 4.4.9 can be used for the general approach to privacy-preserving geofencing, specifically for service evaluation of area sizes and cell quantity. The second part is realized by the SPM variant using the Geohash cell representation. The following two subsections give examples

of S2 and Geohash representations. Also, additional symbols extending the established alphabets and character-based S2 cells are presented.

5.3.8.1 S2 extended alphabet and re-encoding

Since the SPM protocols are based on string-matching algorithms, it is necessary to make additions and changes to the representation of S2 cells and Geohashes. The original alphabet for an S2 cell consists of only two characters.

$$\Sigma_{\text{S2-cell}} = \{0,1\}$$

An extended alphabet for the arbitrary-shaped geofences representation consisting of individual cells is as follows.

$$\Sigma_{\text{S2-geofence}} = \{0,1,[,],\&\}$$

The additional characters ([,],&) stand for the following.

- [indicates the inclusive start of a cell value, e.g., [0101001001...
-] denotes the inclusive end of a cell value, e.g., ...0101001001]
- & specifies the binding value between two cells within one geofence, e.g., ...011001]&[01000100...

If a symbol is needed to indicate that a text *t*, a combination of cells forming a geofence, can consist of several geofences, i.e., different areas, the following characters can be added to the alphabet Σ. The term *text* is used to be consistent with the wording of the SPM protocols. One party holds the *text t*, and the other party presents the *pattern p*. Letters A, B, C, ... indicate that one cell belongs to the same geofence area, e.g.,

[A[010010011...]]

[B[0011010010...]].

They would be connected using the & characters, e.g.,

&[A[010...001]]&[B[001..110]]&[B[001...001]]&[C[001...111]].

The extended alphabet is based only on 7-bit ASCII characters to ensure low-level implementation capabilities. Before the protocol is executed, the two parties must agree on an alphabet for the *text*, *pattern*, and *wildcard*. The wildcard must not be a character of the alphabet.

S2 re-encoding As an alternative representation of S2 cell values, the following re-encoding is used for use cases instead of a Geohash representation. Using a letter-based representation, similar features can be achieved. This is specifically important for private pattern matching protocols.

1st - 3rd digit (one face of the S2 cube), the six numerical values are used.

$$000 = 0$$

$$001 = 1$$

$$010 = 2$$

$$011 = 3$$

$$100 = 4$$

$$101 = 5$$

4th digit, two characters are used.

$$0 = @$$

$$1 = !$$

5th - 64th digit (always two digits represent the four children), only characters are used.

$$00 = a$$

$$01 = b$$

$$10 = c$$

$$11 = d$$

The alphabet for the re-encoding of the S2 binary representation is

$$\Sigma_{\text{S2-cell-re-encoded}} = \{0, 1, 2, 3, 4, 5, @, !, a, b, c, d\}.$$

Level 0 example, always a face with a 1, i.e., one of six faces, S2 cell level = 0.

- Original: 0101000000000000000000000000000000000000000000000000000000000000
- Re-encoding: 2!

Level 1 example, always 0 at 4th digit, i.e., S2 cell level >= 1.

- Original: 0100010000000000000000000000000000000000000000000000000000000000
- Re-encoding: 2@*b*

Level 30 example, the last 30 pairs (2 digits) of an S2 cell are replaced by a letter.

- Original: 0100011110101000111101100000010101110101010011001000101000101111
- Re-encoding: 2@*bdcccaddbcaabbbdbbbadacaccacdd*

The number of lowercase letters $|x|$ indicates the S2 level (1-30). If there is no letter present, indicated by the ! in the second digit, the S2 level is 0. This applies to the encoding with numerical values for the face and level 0. The first column inside the example in Table 5.6 represents the S2 level. The second column is the re-encoding using

numerical values for the face and level 0 and lowercase letters for the S2 cell in the last column. The extended alphabet for the S2 re-encoding is as follows.

$$\Sigma_{\text{S2re-encoded-geofence}} = \{0, 1, 2, 3, 4, 5, @, !, a, b, c, d, [,], \&\}$$

In the case of multiple geofences, i.e., different arbitrarily shaped areas, the capital letters A-Z can be added to the alphabet, as mentioned above.

$$\Sigma_{\text{S2re-encoded-geofences}} = \{0, 1, 2, 3, 4, 5, @, !, a, b, c, d, [,], \&, \text{A-Z}\}$$

Table 5.6: S2 re-encoding example

Level	Re-encoding examples
0	2!
1	2@*b*
2	2@*bd*
3	2@*bdc*
⋮	⋮
28	2@*bdcccaddbcaabbbdbbbadacaccac*
29	2@*bdcccaddbcaabbbdbbbadacaccacd*
30	2@*bdcccaddbcaabbbdbbbadacaccacdd*

5.3.8.2 Extendend Geohash alphabet

Following the previous section, an approach based on Geohashes as grid cells is presented below.

The alphabet for the Geohashes is based on the base-32 numeral system, which uses a set of 32 characters, each represented by 5 bits (2^5). The original Geohash system uses the alphabet $\Sigma_{\text{Geohash-cell}}$.

$$\Sigma_{\text{Geohash-cell}} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, b, c, d, e, f, g, h, j, k, m, n, p, q, r, s, t, u, v, w, x, y, z\}$$

The extended alphabet for the entire geofences, which consists of individual cells, contains the characters from the S2 example. There is an additional entry to fill up shorter Geohashes to the eight-character length version. The character + is used to indicate the level of the Geohash that can be checked in the pattern matching protocol. In the example below, a Geohash length of 12 characters is used. The + character can also be added to the S2 re-encoding alphabet $\Sigma_{\text{S2re-encoded-geofence}}$ above. This alphabet uses words, i.e., patterns p, of 32 characters instead of the re-encoded 12 to represent an S2 cell.

$$\Sigma_{\text{Geohash-geofence}} =$$
$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, b, c, d, e, f, g, h, j, k, m, n, p, q, r, s, t, u, v, w, x, y, z, [,], \&, +\}$$

$$|\Sigma_{\text{Geohash-geofence}}| = 36$$

An example of a geofence representation with Geohashes is given in the next line.

$$\&[u33db2p+++++]\&[u33db8++++++]\&[u33d8+++++++]$$

5.3.8.3 Secure pattern matching

The secure pattern matching protocol is based on [229], referenced in [132]. The protocol is based on the additive ElGamal encryption described in Appendix A.2.1. The variant that also supports wild cards is described and used. The protocol is already adopted to serve as the basis for comparing grid cells.

In our case, the substrings $\bar{t}$ of the original text t represent a cell in a geofence. The pattern $p \in \Sigma^m$ means either the parent values of a position or a pattern that tests the level of the cells in a geofence.

The text $t \in \Sigma^n$ represents the geofence. Each substring in the text is 14 characters long and starts with a [and ends with a]. The & character is the identifying an initial character of each substring, i.e., a Geohash cell with a level indicating character +. $|+|$ = {0...11}. The substrings start at the positions i = i ×15 for i = {1...#&}. #& is the number of geofence cells in the text t. The length of t is 15 ×#&. The described variant already includes wildcard capabilities in the pattern p.

Each character in the alphabet will be represented by an integer number starting at 1. A wildcard * character will be represented by 0. Each substring $\overline{t_i}$ at position i, i.e., cell in a geofence, is represented by the function $h(x)$.

$$h(x) = \sum_{i=0}^{m} x_i \times |\Sigma|^i \tag{5.63}$$

In the wildcard version, the participant that owns the pattern replaces the characters with a 0 value. The participant sends a *selection vector sv* of length m in encrypted form to the participant that owns the text.

$$sv_i = \begin{cases} E(0) & \text{if } P_i = * \\ E(1) & \text{otherwise} \end{cases} \tag{5.64}$$

This vector is combined with all substrings resulting in a null value at each wildcard position of the pattern. The resulting substrings are represented by $\hat{t_i}$. The protocol continues with the steps that are also present in the exact match variant. The pattern p and each substring $\bar{t}_i$ is encrypted with the additive homomorphic ElGamal function $E()$.

$$p' = E_{pk}(\sum_{i=1}^{m} P_i * |\Sigma|^{i-1}) \tag{5.65}$$

$$\overline{t'}_i = E_{pk}(\sum_{j=o}^{m-1} \hat{t}_{i+j} * |\Sigma|^j) \tag{5.66}$$

For the comparison of the individual patterns (p' and $\overline{t'}_i$), both parties compute all deltas

$$\Delta_j \leftarrow \overline{t'}_i * (p')^{-1} \tag{5.67}$$

which, due to the additive homomorphic ElGamal encryption, results in the value 0 if the two patterns match. The element $(p')^{-1}$ is the additive inverse of p, i.e., its negative value. The *additive inverse* of a number x is the number that results in zero (0) when added to x. It is also called *negation*.

5.3.8.4 Secure substring query for area limitation

The wildcard variant can be used to check the levels of the geofence cells and thus determine the upper boundary of the area to be used in scenarios where the location of the geofence areas is hidden from the server so that the client cannot cover an entire city or country with a given privacy geofence. In this approach, the wildcards will be placed at the positions describing the geofence cell, and the + signs in the Geohash version will be indicators of the pattern being checked against the text, i.e., the geofence.

Examples for patterns p, representing parent cells with three different levels, and a text t representing a geofence.

$$p = \&[******++++++]$$

$$p = \&[*****+++++++]$$

$$p = \&[****++++++++]$$

$$t = \&[u33db2++++++]\&[u33db+++++++]\&[u33d++++++++]$$

This can also be used for traffic congestion markdown in the context of carsharing applications. The traffic congestion is the *text*, and the current location is the *pattern*. Thus, the app and the car holding the pattern learn in a verifiable way that they are in a congestion.

5.3.9 Privacy policies

In the doping control use case, a DCO needs to know the whereabouts of a particular athlete, i.e., the target's location is requested reactively. Considering the DCO as the adversary, one can design an interface in which the athlete's location is presented to the DCO on a need-to-know basis. This includes contextual information about the distance between the DCO and the athlete, the DCO's mode of transportation, and the time of day. This section describes how a policy-based interface can be designed to constrain whereabouts information.

5.3.9.1 Notions of geofences

In this use case, geofences are created by the user. If the individual is within a pre-defined privacy area, a *hierarchical* general polygon is returned to the requesting party, including a private message. Such a hierarchical polygon can be a postcode area. A 5-digit postcode area with a 5-level granularity is used in Germany. In the underlying use case, these

polygons express a general geofence that is semantically independent of a location. The German postcodes divide the areas according to the underlying administrative structure of federal states, cities, and districts. The hierarchical structure makes it ideal for the given context, see Section 7.3.1. The postcode structure can also be used for a distance evaluation between a requesting entity and a target.

Three different notions of geofences are defined. Some geofences apply only to a single user, i.e., *privacy geofences* or *privacy gardens*. Some geofences apply to a group of users, i.e., *general privacy geofences* or *general privacy gardens*. There are areas where it is *not allowed* to define privacy geofences. In the doping control use case, such regions may be determined by organizations that initiate doping controls. These can be areas such as stadiums or training grounds. They may also be transportation hubs, such as airports and train stations, or medical facilities associated with anti-doping or doping control procedures [38] [37] [230]. In Section 6.3 of the proof of implementation chapter, examples of these different notions are presented.

5.3.9.2 Location hierarchy and semantics

As discussed in Section 4.3, one of the most common approaches to ensuring location privacy is to obfuscate a position using a granular location around that point. Typically, most approaches to ensure privacy consider only the client's location.

In the doping control use case, the relative location of a DCO to the athlete is also taken into account. The DCOs only need to know *the* whereabouts that help them perform *their tasks*. Since the location request is reactive, a DCO does not know the location of the requested device in advance. If a DCO is not in the same city as the athlete, the whereabouts information can be obfuscated. For example, suppose an athlete is located in Munich, and the DCO is in Berlin. In the case of a location request, the DCO does not need to know the exact position, i.e., the GPS coordinates, because the doping control is not likely to occur soon. In this case, the information Munich and the displayed geofence around the Bavarian city is a piece of information that the DCO receives when making her request.

A fine granular system can be considered in which the various districts of a city, such as Berlin, are used as geofences. The two individuals would need to be very close to each other to reveal a GPS position. Otherwise, the information is presented along with the postcode boundaries. During the discussion of privacy area ideas, it was noticed that the granularity of the geofence hierarchy needs to be further evaluated, see Figure 5.23.

If an area is marked as a (personal or general) private geofence, the DCO interface would inform her that the athlete is within a private area and displays the same obfuscated postcode area in a different color.

Examples of the interfaces can be found in the implementation chapter, Section 6.3. The evaluation chapter, Section 7.4, provides an overview of the user feedback collected during the research project.

5.3.9.3 Area limitation using an interface

In a non-cryptographic solution, the target user's interface limits the creation and properties of the geofences. The screenshot below exemplifies a minimum level of 20, a

	Athlete's Location	DCO's Location
Country	Germany	Germany
State	Berlin	Berlin
City	Berlin	Berlin
City District	Mitte	Tiergarten

Figure 5.23: In the athlete whereabouts use case, the doping control officer requests a position to conduct a meeting and a doping control. The athlete's position is returned to the service, and the DCO's position is used to calculate the relevant location information. If the distance between the two parties is significant, the athlete's position can be obfuscated, e.g., by using different levels of postcode areas. In the figure, the two parties are located in different districts of Berlin. Therefore, only one postcode area is returned to the DCO as location information. If the DCO moves to the same district, a new request will yield the exact location of the athlete. If the DCO moves further away from the athlete, the information is less accurate, e.g., only the city level is returned instead of a federal state. For example, if the DCO requests a position from within Potsdam, only Berlin is returned as the athlete's location information.

maximum level of 22, and a maximum number of 50 cells [139]. The left circle (small) follows these constraints, and the right circle (large) does not.

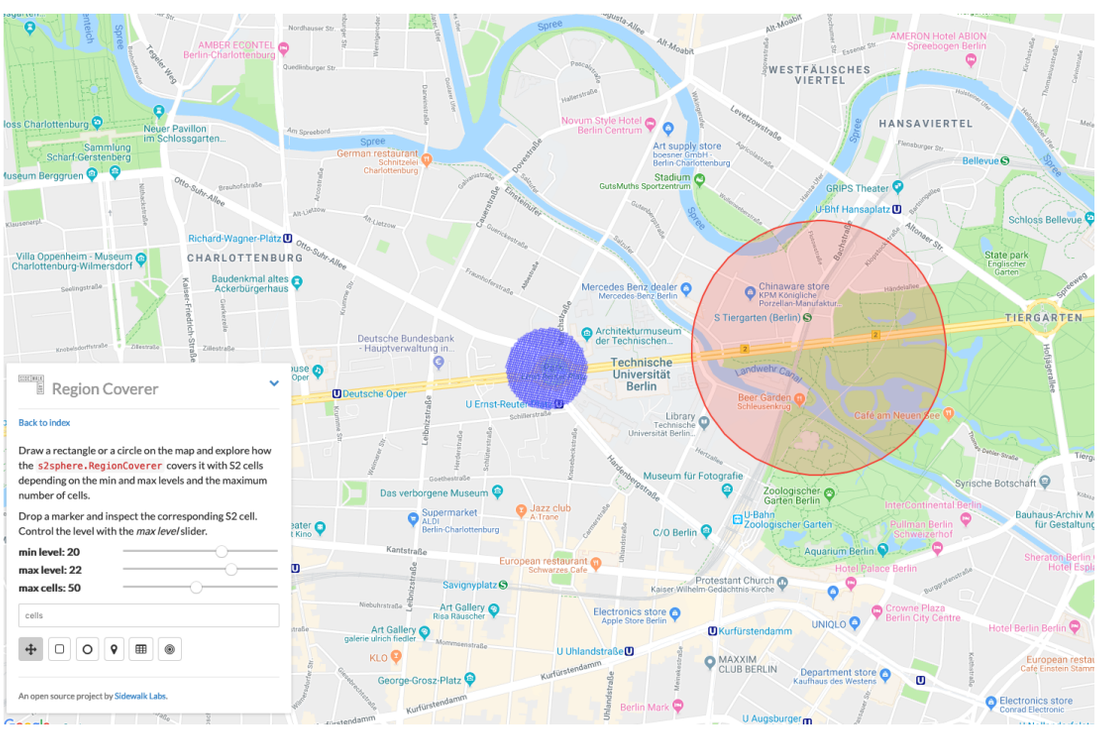


Figure 5.24: During the setup phase, the target interface could limit the size and number of geofences. The figure shows an example using the Sidewalk Labs service. The implementation chapter explains the project interfaces for the Whereabouts use case.[5]

In Section 6.3, the interfaces developed in the PARADISE project for the whereabouts use case also show examples of the policy-based approach and limitation settings.

[5]Image sources: Screenshot of the http://s2.sidewalklabs.com/regioncoverer/ website.

5.4 Enabling additional features and use cases

This section provides an overview of functionalities enabled by the presented protocols and encodings to support additional use cases and requirements. The most important functionalities are privacy-preserving *proximity testing*, *contact tracing*, *speed measuring*, *traffic congestion evaluation*, and *geofences on distributed ledgers*.

5.4.1 Privacy-preserving proximity testing

The encodings and protocols can be used for privacy-preserving distance measuring, i.e., it can be reliably evaluated whether target A and target B are within a pre-defined distance of each other. The method used for this particular function is based on the *grid cell hashing* approach. In Figure 5.25, two S2 points (small circle) represent the positions of A (blue) and B (green). The parents (red) of A and B are calculated. The S2 cell level of the same parent (large red quadrilateral in the left example) provides the maximum range the two participants are located to each other. Calculating the maximum distance of the two S2 points within this S2 cell will approximate the maximum distance between A and B.

This can be illustrated using the S2 cell measurement statistics from Section 4.2. The example is given for n=2 target users. Both mobile devices compute the S2 points of their current location. For each S2 point, all 30 parents are calculated on their devices. Both 30 values are hashed using the same hash algorithm. All 60 values together are sent to the service along with their respective S2 levels. If there is a match, the S2 level information is used to look up the average edge length of that particular level. An S2 cell at level 15 has an edge length between 212 and 306 meters. From this value, the two target users within the cell can be considered and are, at most

$$\sqrt{212m^2 + 306m^2} \approx 372m \tag{5.68}$$

apart, with each area approximated as a rectangle to simplify the calculations since it should not be evident in which area of the six faces of the S2 cube the targets are located.

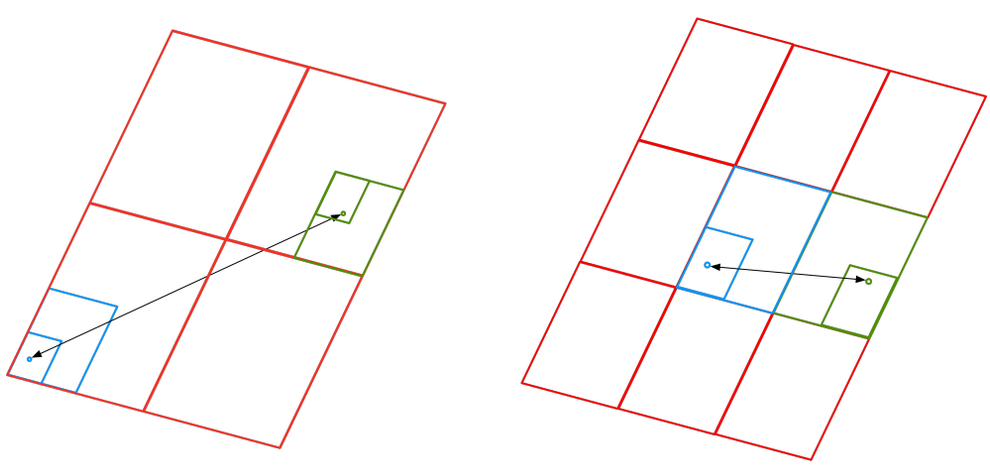


Figure 5.25: Two examples of a privacy-preserving proximity test using S2 geocoding. Two dots (green and blue) and their parent values in green and blue. The red areas indicate parent values that have either a common parent or a parent that is a neighboring cell.

The use of the same hash function on each target device of A and B underlines the privacy

aspect of this approach. This can be cryptographically hardened by using an additional salt value. In this approach, there is no need to store a geofence or other values with the service provider, as only the client values are compared along with their S2 level details. Using different salt values for different user groups would also provide more flexibility in terms of supported use cases, e.g., a location-based sports network could categorize different forms of sport or cities and their respective user bases.

The right side of Figure 5.25 shows a slightly different approach to testing the distance between two S2 points. The largest blue parent quadrilateral, including A, has eight neighboring cells of equal size. The green adjacent cell includes the S2 point of B. The distance can be measured by calculating the maximum distance of such two points, using the S2 cell statistical edge lengths of two cells, i.e., parent of A and neighbor parent of B, of the respective cell level.

This *simple* mechanism is also not considered as privacy-preserving as a version in which n users would share a salt value. Figure 5.26 shows a sequence diagram for this approach. Using the parent-neighbor strategy for proximity testing would also be usable with other approaches.

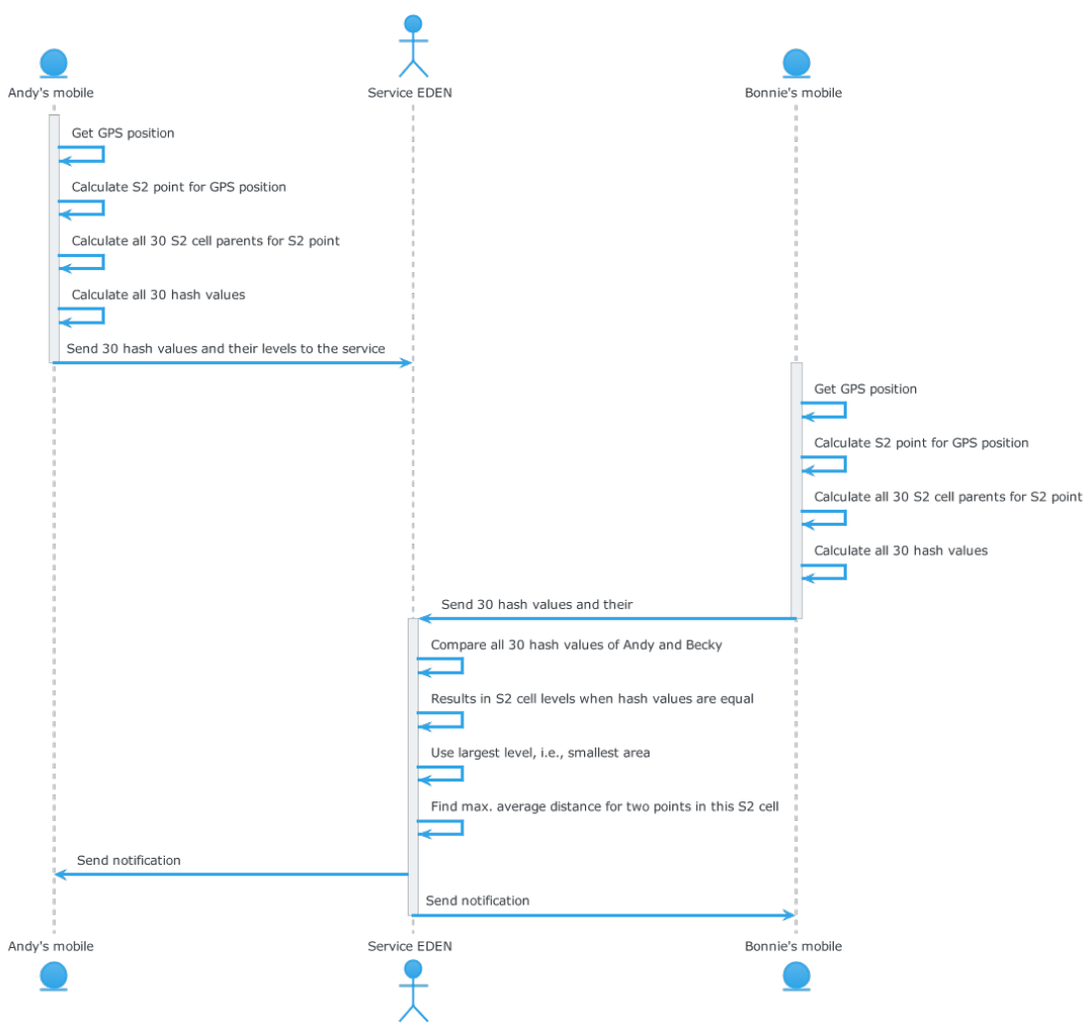


Figure 5.26: S2 grid cell distance approximation sequence diagram showing a service evaluation of two targets.

5.4.2 Privacy-preserving velocity measuring

The combined knowledge of when the client updates its encoded parent values and the change in the level indicator can indirectly estimate the approximate speed. Certain level values of the encoded S2 point will change with every update as the target moves. By knowing the S2 cell statistic and the minimum level of the parent that changed to the last position update, the velocity can be determined. The average area of a level 19 cell is 309 m^2, which corresponds to an idealized edge length of a square of 17.6m. The diagonal of this square is 24.9m. This is the maximum distance the target client will travel before the parent cell-level changes. A target client that changes its encoded parent value at level 19 every second has an average speed of 24.9 m/s, i.e., $\approx$ 90 km/h. If the maximum cell area is used, the velocity is 24.9 m/s and 19, 2 m/s for the minimum cell area. The minimum, average, and maximum target velocities can be assumed with this approximation. A target for which the higher-order values do not change is considered stationary. It should be emphasized that the actual grid cell values are not needed if the minimum, average, or maximum cell areas are used for speed measurements. This can be considered a *privacy feature* when using the grid cell hashing approach presented in this thesis.

5.4.3 Dynamic geofences for traffic congestions

As described in the cryptographic accumulator section, geofences can be dynamic, and evaluating the target position against them must not violate the privacy-preserving nature of the solution. Dynamic geofences can be used if carsharing companies want to offer different minute rates of their service when a user is stuck in a traffic jam. In this case, the traffic congestion is represented by a changing geofence and is only valid for a shorter time. The type of CAs from Section 5.3.5 seems to be an appropriate approach that allows this type of privacy-preserving evaluation. It must also be reiterated that the parties will follow the protocol in the semi-honest advisory model. However, it is evident that such a solution has its limitations since a city with only one traffic jam at a given time would indirectly reveal the target's location if the user can be associated with that city through her address or pickup location, which is known in the case of a carsharing service. Also, the variants of the Bloom filters from Section 5.3.2, which provide similar functionalities as cryptographic accumulators, should be considered as an approach to enable dynamic geofences.

5.4.4 Pandemic health information

The recent introduction of virus-related warning apps based on several approaches, including privacy-preserving client evaluation rather than server-based, raised how to provide additional information to national health services to better understand the dynamics of a deadly pandemic. This section gives an overview of how the previously described protocols and schemes can be adapted for privacy-preserving service evaluation. The focus is on the location and positioning aspects of target clients.

The answers to these questions include valuable information for national and international health services.

- Where did the potential infection happen, i.e., in which area?
- How many people were in close proximity?

- Was the infectious event indoors or outdoors?
- Was it on the grounds of a school, workplace, or shop?
- Was it a potential super-spreading event, such as a wedding, a birthday party, or an open space office?
- Did the target stay in an area of residence or another state or country?
- What mode of transport did an infected person take?
- Where did the infectious person travel?
- How many additional people did the contagious person meet since then?
- How long were the exposure times?
- What type of smartphone transmitted the values, i.e., sent and received?

Current contact tracing apps use shared IDs with transmitted power levels to determine if two targets are close to each other and thus at risk of infection. These IDs are evaluated on the client, making it privacy-preserving since no server component computes the results. In its basic form, this is a set membership test, i.e., it checks whether one or more IDs from one client list are also included in another client's list. The privacy protocols presented evaluate set memberships on the service side at its core. Therefore, one can solve the above challenges by defining multiple geofences, sharing personal values, such as keys between clients, and defining on-the-fly areas related to temporary events or transport routes and vehicles. These areas represent postcodes, event locations, school grounds, university campuses, train networks, or office building complexes. The service evaluates information provided by individuals on how many risk encounters have occurred in these areas to identify risk factors for the specific site characteristics. These risk factors help researchers, public health authorities, and policymakers to assess the effectiveness of countermeasures in order to adjust and optimize them.

5.4.5 Distributed ledger technology

In Victor and Zickau [121], it was shown that grid cells can specify geofences within smart contracts [61] [62]. Combined with a supplied position, the contract can determine if the position is within the boundaries and trigger an action accordingly. The implementation uses a mobile-network-based location oracle to provide real-world location information, see Section 6.4. Several interaction strategies between the oracle and the smart contract are provided with an algorithm that compares a position with one or more geofences. The grid cell approach in distributed ledger environments is initially described in this subsection without focusing on privacy, but it can easily be extended to include the hashing variant. In the cited article, a distributed-ledger-based carsharing use case was described to motivate the requirements and the evaluation in Section 7.5.

5.4.5.1 Geofences in smart contracts

A hierarchical grid-based geo encoding such as Geohashes or S2 cells is used to represent and query geolocations in smart contracts because a point-in-polygon calculation can be performed with little computational overhead. The result of this process is a set of identifiers, where each identifier describes a subset of the geofence as shown in Figure 5.27.

Each cell identifier is stored in a key-value map within a smart contract, with each defined value set to True(1). The default value for undefined keys is to return False(0). The key-

value map can be queried to determine if a specific cell is included in the geofence. If the exact cell was used to define part of the geofence, a single lookup would return a positive response True(1).

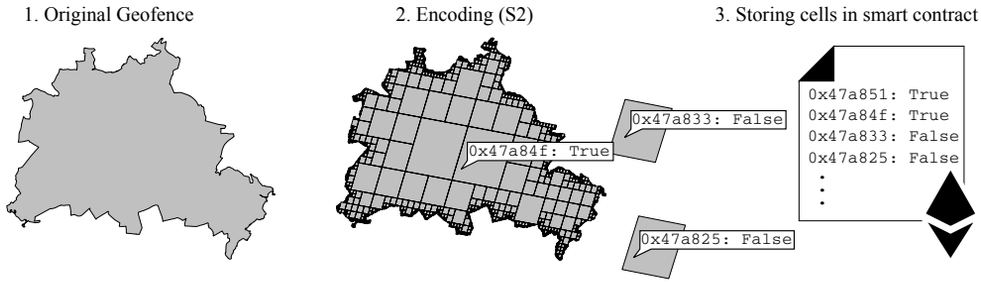


Figure 5.27: A geofence (1.) is encoded with S2 grid cells (2.) and stored in a key-value map within a smart contract (3.). Undefined and invalid keys, such as 0x47a825, are false (0). The S2 hex token representation is used in the distributed ledger-based use case.[6]

A single query would always return a result in the naive scenario that uses the same cell size for the entire geofence. However, since the geofence may be defined with different cell resolutions to reduce the number of identifiers needed, it may also contain some large cells. Increasing the size of the cell by gradually decreasing its resolution level step by step is necessary until a match is found, see Table 5.1, or the lowest geofence resolution has been exceeded. This approach allows the determination of set memberships without (geometric) calculations.

5.4.5.2 The mobile network operator as a location oracle

The smart contract must be supplied with reliable target updates to perform location tracking. In the approach presented, the existing infrastructure of a mobile network operator was used. Typically, such an operator has a large network of cell towers. At the same time, a target client must be locatable with respect to the network cell. A client can report its cell location with a *location update*. The network can also query the current cell using *paging*. The sample network operator acts as a location oracle with the existing mobile network protocols and infrastructures.

One shortcoming may be that an operator can only provide information about a terminal's location within a covered area. If the device is *roaming*, positioning may not be possible. In this case, several mobile network operators (MNO) could cooperate to be a single oracle covering, for example, all European countries and could therefore be used in international tracing and tracking use cases.

5.4.5.3 Resolution alignment

For the cell coverage areas to match the geofence definitions in the smart contract, an encoding process must be applied that includes the choice of area resolution. As shown in Figure 7.46, the process can result in a set of multiple grid cells. Although this sometimes significantly expands the coverage and thus the position claim, it is only an approximation that can be beneficial to network operators who typically want to keep information about the exact cell coverage areas confidential.

[6]Image source: Figure is from the original publication, Victor and Zickau (2018), and was collaboratively created by Friedhelm Victor and Sebastian Zickau.

An important consideration is the alignment of encoding resolutions between geofence and network position estimation. The maximum grid cell size used in the position estimation determines the smallest grid cell size of the geofence. This has a direct impact on the accuracy of the geofence encoding.

5.4.5.4 Oracle-contract interaction strategies

The location oracle operated by the mobile network operator is a regular distributed ledger participant that sends transactions to positioning smart contracts when adequately set up. The initial configuration is as follows,

- a target client ID that is tracked,
- at least one smart contract address that is to be called via the position,
- the maximum cell-level resolution of the defined geofence, and
- an interaction strategy that describes when new location information should be transmitted.

The geofence's smallest resolution (maximum cell level) must be defined in advance since the oracle should only transmit larger grid cells. If it were to transmit these, it would not be possible to say with certainty that the mobile terminal is inside the geofence. As for the interaction strategy, there are three approaches. The position of the terminal can be transmitted

- *at regular intervals (proactively and reactively)*: This has the benefit that a complete position history is created and can be traced at later points in time. If the client does not send position update messages to the network regularly, paging requests may be required, potentially creating a significant signaling overhead. Nevertheless, a high number of distributed ledger transactions may occur, resulting in increased storage costs.
- *to events triggered in the smart contract (reactively)*: For example, when a shared car is returned, a smart contract might want to ensure that it is parked within the business area. In case of a conceivable driver deposit recovery, the oracle could transmit the location.
- *to a moving distance* (proactively): to reduce the number of updates sent to the contract, the oracle can monitor the moving distance and provide an updated position estimate only when the target has moved a predefined distance.
- *on geofence violation only* (reactively): to minimize the number of transactions initiated by the oracle, it could monitor the terminal's position off-chain and transmit the location only in case of a geofence violation.
- *on a combination of the above (reactively and proactively)*: a position update may be desirable during the initial rental process. After that, periodic updates can be applied.

For the evaluation, the carsharing scenario was chosen, but other use cases can be implemented in the proposed way in the context of DLTs and smart contracts. Section 6.4 in the proof of concept implementation chapter shows a snipped of the DLT-based algorithm.

5.4.5.5 Decentralized carsharing

Unlike existing blockchain-based carsharing applications, a truly decentralized approach would allow users to set up their smart contracts to specify their requirements without relying on a central contract or platform to dictate terms and take commissions. To facilitate individual carsharing restrictions, the car owner can define a geofence for a city like Berlin, perhaps including or removing certain regions. Other time and deposit constraints are conceivable.

A potential renter would search for any smart contracts that offer carsharing and look at their current locations and rental terms and conditions. Ideally, the smart contract would support utility functions such as geofence inspections and rental – where a deposit could be included in the contract. To verify that the location oracle is working correctly, the user could query the car's location and confirm the result in the real world. While driving, the car's built-in terminal enables periodic location updates via the cellular network, which are sent to the smart contract. When the user leaves the geofence, the deposit is debited.

5.5 Possible system integrations

The approaches presented address information exchanges between a client OS, a mobile application (MA), a network operator (NO), and an LBS provider to comply with data protection requirements. Before using a MA, the target user (TU) can create an account. It should be emphasized that the majority of relevant approaches do not consider a trusted third party (TTP).

To integrate the data protection approaches discussed, it is necessary to distinguish at which service or application level this integration can or should take place. This section discusses how the proposed approaches can be integrated into existing infrastructures, for example, into the

- target client operating system (OS),
- mobile application (MA),
- LBS provider (LBSP),
- third-party software (TPS), such as databases,
- network operator (NO),
- edge entity of a network operator, and
- GNSS operator.

In Section 3.2, additional roles and actors were defined but are not covered in this subsection.

5.5.1 Existing infrastructures

The privacy setting capabilities of client operating systems vary depending on the OS manufacturer (OSM) and version. Some include binary settings for specific types of data. They provide the ability to share an exact location with the service only once (introduced in Apple's iOS version 13 in late 2019). This behavior or setting option should be extended to enable accurate LBS evaluation and protect user location data from a work perspective.

To protect the exact position of a TU, the OSM's API for mobile application developers

(MADs) must be extended with privacy-preserving methods. A TU should have the choice to enable privacy-preserving geofencing evaluation. In the wake of the Apple iOS 13 launch in 2019, MADs saw their business models in jeopardy and called it anti-competitive [231]. This example shows the competing interests of the parties involved. The OSM wants to provide privacy to the users as a part of a business strategy and prevent MADs from relying on revenue from selling location data through free-of-charge apps.

In the semi-honest model, an LBS service is considered honest but curious. That means that any information the service can obtain from either the API/OSM or the network operator is cross-referenced with the user's identity and data to gain knowledge about private information. Therefore, the OSM must restrict access to information the service might use.

As of 2021, there are economic incentives for both the operating system (OS) developers and service developers to access any information they might gain about a target user. Only in recent years, with the users' broader understanding of privacy-violating applications, the OS manufacturers have offered more and more privacy settings within their mobile OSs.

5.5.2 Service database

Most approaches rely on a setup where the service and client exchange data to evaluate the position at a later point in time. Following the example use cases, the service needs to know or not know the following information.

- User identifier
- Additional user information, such as home and work address
- Mode of transport
- Geofence location and semantics
- Group identifiers and information

This information is detailed below with example use cases and related applications and summarized in Table 5.7 for the seven use cases.

User identifier and information If the geofences are unique to each target user, the service must store an identifier with the values that represent the geofence(s). In the carsharing use case, the users must be identified to be charged for the service. In the whereabouts use case, athletes must also be identified.

Mode of transport In the cycle path use case, the individual users do not need to be identified to evaluate their paths. However, in this specific case, it is necessary to know the mode of transportation, e.g., cycling or riding an e-bike or an electric scooter. This also applies to the toll charge use case where a target user/vehicle may be considered private. Still, specific details about the vehicle may be known to the service to apply different tolls for different cars.

Geofence semantic If the values represent very private locations of the user, the service should have no way to identify the locations the values represent. This is true for the *whereabouts* use case, where athletes mark areas they consider private. In the carsharing example, the geofence is known to both the user and the service. The users need to know

where they can drive or park or in which areas additional charges may apply when the driver leaves a state or district. In the cycle path evaluation, the same arguments apply to all parties who know the area characteristics.

Grouping Multiple users may share locations and places near them in the social network use case. Nevertheless, the privacy of their meeting places must be hidden from the social network service. Social groups can be considered dating groups, sports groups, or people with common interests. The service does not need to know where these people live and meet.

A question that emerged during the 2020 pandemic asked at what places and locations the chances are higher of spreading infections. To answer this, one can use geofences with specific semantics, such as *educational grounds and kindergartens*, *office spaces*, *public transport stations*, *industrial and manufacturing sites*, *sport and leisure areas*, and *restaurants*. If red-flagged alerts of possible infection come up, the system would evaluate its location against these predefined areas to calculate where most close contacts are happening.

Table 5.7: Use cases and service knowledge

	User identified	Add. user information is known	Mode of transport known	Service knows geofences	Target user is in an (anonymous) group	Target user is not in a group
1 Athlete's whereabouts	●	●	○	○	○	●
2 Carsharing	●	●	●	●	○	●
3 Cycle path evaluation	○	○	●	●	●	○
4 Remote tolling	◐	◐	●	●	○	●
5 Social network distance	●	◐	○	○	●	○
6 Contact tracing	○	○	◐	◐	◐	◐
7 Telemetry data	●	●	●	○	○	●

● Statement is applicable to use case
◐ Statement is applicable to use case variation
○ Statement is not applicable to use case

5.5.3 Location service provider

Third-party location services, such as Skyhook [232], operate independently of the OS and device manufacturers but provide services to them. Skyhook, founded in 2003, started to geolocate Wi-Fi access points through a technique called wardriving. Vehicles drove through urban locations to map Wi-Fi hotspots and combined their information with GPS positions. An app has access to the Wi-Fi access point information, i.e., the service set identifier (SSID), in addition to MAC addresses and encryption types. Using an SDK to query the exact location of the access point may not need other positioning technologies, like GPS.

Other such services are WiGLE [233], Radio Cells [234], formerly OpenBMap, OpenCellID [235], and Geomena [236]. Some of these services using crowdsourced data also include cell tower information about mobile cell towers and Bluetooth IDs, with the same intention of locating devices and enabling LBS applications.

5.6 Chapter summary

The main chapter of the thesis started with a layered overview of the general concepts, the privacy and security differences, and an argument for the forthcoming cryptographic approaches. The layered concepts structured the remaining chapter. It started with presenting the geo encoding systems used and how they support the set membership evaluation approach used later. The ten different cryptographic procedures were all detailed in their respective subsections. These approaches also enable additional use cases and features, as presented. The chapter concluded with a short assessment of integrating the conceptual work into existing service and provider infrastructures.

Well, nobody's perfect.

Some Like It Hot (1959)

6 Proof of Concept Implementations

The concepts presented and described in Chapter 5 have been partially implemented to provide proof of concept and design reference implementations, as well as an evaluation of those. This chapter describes individual implementation details and procedures for building and installing individual projects, including their configuration and operation. First, a description of the evaluation setup for *cryptographic hashing*, *homomorphic encryption*, and *zero-knowledge proof schemes* is presented. Additional descriptions of two *alternative ZKP libraries* and geofences within *DLT environments* are detailed. Furthermore, user interfaces of an *ABE service* and a corresponding *mobile application* and a browser-based *whereabouts demonstrator* are illustrated.

6.1 Evaluation framework

The described evaluation framework was implemented in the context of a supervised master thesis by Ersin Bayraktar in 2019. This framework includes hashing, homomorphic encryption, and zero-knowledge proof schemes. The results are presented and discussed in Chapter 7 together with additional schemes and their correlation with the 17 objectives.

6.1.1 Architecture

A flexible but robust evaluation framework was designed to evaluate the three schemes. The framework uses Docker containers and provides REST-API endpoints for loading the (location) data. Both client and server agents are simulated inside separate containers for every cryptographic scheme. An interface implementation for every scheme is provided for building and executing tests. As the scheme capabilities are different, it is possible to provide further tests for each scheme. Some schemes can process large amounts of data, while others are not. This is reflected in the computational result figures in the evaluation chapter. Since dependencies, environments, and configurations for each scheme are different, the usage of Dockerfiles allows the decoupling of tools from the evaluation

framework. The evaluation framework is also responsible for generating geofences and location data and transferring them to containers respectively.

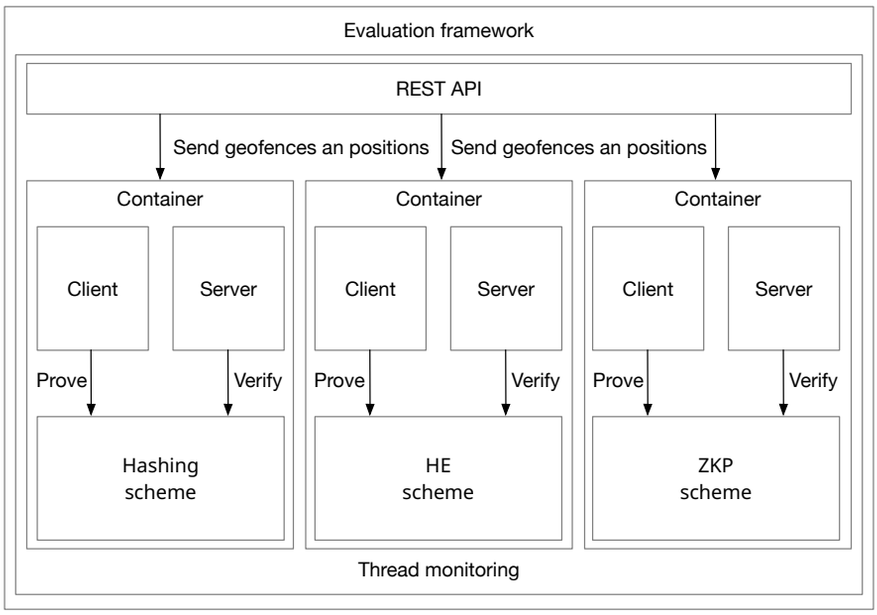


Figure 6.1: The workflow and components of the evaluation framework are abstracted. It comprises three containers for the hashing, HE, and ZKP procedures. They each contain a client and a server component. The REST API provides the corresponding location data to the containers. Global thread monitoring records the computing times for the build, proof generation, and verification phases.[1]

The evaluation framework is also capable of monitoring threads. Docker statistics for memory consumption and processing power are logged continuously by using monitoring threads. Besides the Docker statistics, build logs, container logs, and application logs are also recorded.

As shown in Figure 6.1, the workflow steps of the evaluation framework are as follows.

- An application is picked.
- The corresponding Dockerfile is chosen.
- The Docker image is built and run.
- The application communicates via exposed ports and its REST-API.
- The evaluation framework communicates via its REST-API with the tools.
- The Geofences inside the test cases are loaded via the endpoints.
- The communication payload is serialized as JSON objects and de-serialized inside the endpoint handlers.
- To represent a coordinate, a list of S2 cells and Geohashes corresponding to the position are transferred and resolved. To have all precision levels for particular coordinates, lists of S2 cells and Geohashes are sent. The schemes compute whether the position resides inside any of the geofences. If it is inside, a geofence label is returned, a `not_found` if it is outside.
- After running all test cases for a scheme, the container corresponding to that scheme is destroyed, using the `docker_kill` command.
- The same steps are applied to the subsequent scheme.

Pequin [237] and NEXUS [176] were extended to run the described test cases among the

[1] Image source: The original version of the figure was created in the context of the supervised master thesis.

four schemes. Initially, they only supported either perfect rectangular or radius-based circular geofences. Ersin Bayraktar implemented the NEXUS parallel and the grid cell hashing schemes as part of a supervised master thesis.

6.1.1.1 Main application and template interface

The evaluation framework has a `MainApplication` module. It is responsible for running the four schemes with their respective test cases. To be able to run a scheme, predefined rules are in place. An abstract class `AppTemplate` is created, see Listing 6.1, that all schemes implement. `AppTemplate` has seven methods. Two are common to all schemes, and five are scheme-specific. The `__init__` and `__def__` methods (Lines 7 and 51) are constructor and destructor. Inside the constructor, the scheme-specific loggers are initialized. If container fragments are left, they are cleaned, and a new container is built and run. After a container starts, container log streamers are configured to be used by the `MainApplication`. With the destructor, containers are stopped and removed.

The shared methods are evaluation-specific methods and are implemented by each scheme. These methods include `prepare`, `execute_client`, `execute_server`, `get_name`, and `get_test_cases` (Lines 31 to 45). The `prepare` method loads the geofences. `execute_client` sends the position and generates a client proof. `execute_server` verifies the proof and gets the response, i.e., a label or a `not_found`. `get_name` is used to get the scheme name. `get_test_cases` is used to get the test cases for the corresponding scheme. Furthermore, the `prepare`, `execute_client`, and `execute_server` methods return their precise execution times.

Besides these methods, in `AppTemplate`, some variables are specifically set for each scheme. These are `execution_path`, `img_tag`, `client_port`, and `server_port`. Each scheme sets its execution path separately, as using the same Docker image tag may cause the loss of container caches. Container caches are important, especially for Pequin, since a build can take more than one hour. The cache could be lost if a different scheme used an identical tag. To test endpoints residing in containers, exposed client and server ports are used. Even though all scheme containers are destroyed after their execution automatically, to mitigate any exceptions, different ports are utilized, as these should be defined not only for exposing the port while running the container in the constructor but also to call the endpoints while running the tests.

```
1  class AppTemplate(ABC):
2      execution_path = None
3      img_tag = None
4      client_port = 0
5      server_port = 0
6
7      def __init__(self):
8          self.app_logger = logging.getLogger(self.get_name())
9          self.app_logger.addHandler(logging.FileHandler(self.get_name() + "_output.txt
       ", "w"))
10         self.app_logger.setLevel(logging.DEBUG)
11         self.client = docker.from_env()
12         print("docker image is building")
13         self.image = self.client.images.build(path=self.execution_path, tag=self.
       img_tag)
14         self.stop_leftover_containers()
15         port_dict = {'%d/tcp' % self.client_port: self.client_port}
16         if self.server_port:
```

```
            port_dict['%d/tcp' % self.server_port] = self.server_port
        self.container = self.client.containers.run(self.img_tag, ports=port_dict,
    detach=True,
                                                    labels=[self.img_tag])
        self.log_stream: Type[CancellableStream] = self.container.logs(stream=True,
    follow=True)
        print("docker image is running")
        sleep(1)

    def stop_leftover_containers(self):
        for i in self.client.containers.list():
            labels = i.labels
            if self.img_tag in labels:
                self.app_logger.info("leftover container with label %s - Cleaning.." %
     self.img_tag)
                i.stop()
    @abstractmethod
    def prepare(self, test_case=None):
        raise NotImplementedError("Should have implemented this")
    @abstractmethod
    def execute_client(self, path: List, test_case=None):
        raise NotImplementedError("Should have implemented this")
    @abstractmethod
    def execute_server(self, test_case=None):
        raise NotImplementedError("Should have implemented this")
    @staticmethod
    @abstractmethod
    def get_name():
        raise NotImplementedError("Should have implemented this")
    @staticmethod
    @abstractmethod
    def get_test_cases():
        raise NotImplementedError("Should have implemented this")
    @staticmethod
    def get_shortened_path(path, count):
        path["location"] = path["location"][:count]
        return path
    def __del__(self):
        try:
            self.container.kill()
        except Exception as e:
            self.app_logger.warning(e)
```

Listing 6.1: AppTemplate interface

The test cases include different positions and geofences. The Cartesian product of these two sets composes the test cases. There are two types of position sets, a driving route from the Ernst-Reuter-Platz in Berlin Charlottenburg to the Rollberg cinema in Berlin Neukölln (as shown in Figure 7.12) and random positions within Berlin. Both of these sets have 400 individual coordinates. Furthermore, those positions are represented by coordinates, S2 cell values, and Geohash values. To represent S2 cells and Geohashes, all their respective levels are utilized. As test geofences, Berlin is represented by its postcode areas, while each area is defined separately. At this moment, not all levels are used. High precision can cause large data sets not suitable for all schemes. The accuracy is utilized from the lowest sensible to the highest processable one, as seen in the schemes evaluation sections in Chapter 7.

The `MainApplication` has several modules to run and monitor schemes. It consists of

three nested for-loops. First, it iterates through all possible schemes. The second iterates through geofences. The third iterates through positions. When the executor methods are called, they return execution times so that the duration values can be logged. To have additional evaluation results, a function called `start_docker_threads` is executed, shown in Listing 6.2. This function starts two different threads, one to monitor the memory consumptions and one to log them.

```
def start_docker_threads(app, instance):
    app_docker_logger = logging.getLogger(app.get_name() + "-docker")
    app_docker_logger.addHandler(logging.FileHandler(app.get_name() + "_docker.txt", "
    w"))
    app_docker_logger.setLevel(logging.DEBUG)
    threads = [
        Thread(target=docker_logger, args=(app_docker_logger, instance.log_stream)),
        Thread(target=run_memory_stats, args=(instance,))
    ]
    for thread in threads:
        thread.start()
    return threads
```

Listing 6.2: Docker thread starter

The first thread is responsible for streaming container logs to the evaluation framework. It is called `docker_logger` as shown in Listing 6.3. It utilizes the streamer of the corresponding scheme container, which is set when the execution object is initialized. Whenever there is a log event in the stream, it is saved into a file.

```
def docker_logger(app_logger, log_stream):
    for i in log_stream:
        app_logger.info(i.decode("utf8"))
```

Listing 6.3: Docker logger thread

The second thread is responsible for monitoring memory consumption. It parses the container stats to memory stats and checks the memory each second. The memory consumption is measured for different phases separately because of the assumption of having different memory usage levels when data is loaded or the proof is generated. The memory monitoring thread is called `run_memory_stats` and can be seen in Listing 6.4.

```
def run_memory_stats(instance):
    global max_mem_usage
    global finished
    max_mem_usage = 0
    finished = False
    container_id = instance.container.id
    client = docker.APIClient(base_url='unix://var/run/docker.sock')
    for i in client.stats(container_id, decode=True):
        if finished:
            break
        mem_usage = i["memory_stats"]["usage"]
        if mem_usage > max_mem_usage:
            max_mem_usage = mem_usage
    print("MEMORY_STAT_THREAD FINISHED")
```

Listing 6.4: Memory monitoring thread

6.1.2 Grid cell hashing

Grid cell hashing was developed without any pre-existing library since no hashing scheme was mainly focused on privacy-preserving location verification. The other schemes have three functionalities: *loading data*, *proof generation*, and *verification*. This grid cell hashing scheme is implemented in the Python programming language.

Loading data The data loading phase occurs on the client-side. All received geofences are hashed using the SHA512 algorithm. This algorithm was chosen due to its robust structure. Having all possible S2 cells or Geohash values as plaintext available, it is possible to pre-calculate objects using SHA512. For this reason, a client-specific salt value is utilized, as described in Section 5.3.1. After hashing all geofence grid cells for all Berlin postcodes, a hash-map containing the postcode as a key and its processed geofence cells as values are sent to the server.

Proof generation Generating proof also happens on the client-side. Compared to the geofence grid cell data, a single encoded position point is significantly smaller. This yields a fast computation. All its possible parent values represent the position as an S2 point or the smallest Geohash value to find a matching cell. All these parents are hashed using the same salt value and sent to the server to be verified.

Verification As described previously, the verification is a simple set membership query. This part only happens on the server-side. Among all pre-defined geofences, the received location is searched for via iteration. The corresponding postcode label is returned when a match is found or the `not_found` response is given.

Containerization The containerization of the grid cell hashing approach is simple. First, the Unix and Python dependencies are installed, then the application is set up and run. The difference from the other schemes is that the grid cell hashing exposes both client and server ports. While for the server, port `6000` is utilized, for the client port `6001`.

6.1.3 Homomorphic encryption

Two homomorphic encryption schemes were implemented and tested with the described evaluation framework, the NEXUS approach, and its parallelized version.

6.1.3.1 NEXUS

NEXUS was developed in the context of a proof of concept implementation of a master thesis and described in [176]. To evaluate it, changes were made to the initial implementation. These changes do not affect the core mechanism. A performance increase was measured during the process. It is written in Python. A REST-API interface with three endpoints is available to communicate with the container.

Loading data To load the geofences, its REST-API `geofence_register` endpoint is utilized. This part differed in the NEXUS implementation, and the change increased the performance. Initially, both the position and the geofences were encrypted for

each received position. In the current implementation by Ersin Bayraktar, the geofence encryption is moved to the data loading part and is triggered only when new geofences are added. Geofences are encrypted by using homomorphic encryption and the client's public key.

Proof generation To generate proofs, both the geofences and position are required. Since the geofences are already loaded and encrypted, only the position is received by the `generate_proof` endpoint. The positions are then encrypted by using the same public key of the client. Here, the Paillier Python library [238] is utilized. Details for the Paillier homomorphic encryption scheme [239] are summarized in Appendix A.2.2. The encrypted geofences and position are subtracted homomorphically to generate location proofs as described in Sections 4.4.3 and 5.3.3.

Verification In the verification part, the encrypted proofs are decrypted to view the result of the subtractions. If it satisfies the condition of a position inside a geofence, the corresponding postcode label is returned. Otherwise, the `not_found` is replied.

Containerization Since NEXUS does not have many dependencies, containerization is straightforward. In the Dockerfile, environment variables are set for pre-installation. Then required Unix and Python dependencies are loaded. The REST-API listens on port `5000` as exposed in the container setup file.

6.1.3.2 NEXUS parallel

Compared to the previously described NEXUS implementation, the performance can be increased significantly through parallelization. Since NEXUS highly relied on an iteration mechanism, it is sensible and achievable to parallelize the data loading and proof generation parts. Furthermore, parallelization is essential in achieving feasibility, considering that many new mobile devices have multiple CPU cores. A trade-off is the corresponding increase in battery consumption.

Loading data As in the NEXUS implementation, via the REST-API `geofence_register` endpoint, positions and geofences are loaded. The difference lies in the iteration of the postcode geofences. They are distributed to each available CPU core to maximize performance. All computation nodes collect and return each output, i.e., the encrypted geofences, inside a collection utilized in the proof generation step.

Proof generation In this step, parallelization of the iterations is also used. First, the position values are encrypted using the same public key used in data loading. Then instead of iterating through all geofences to subtract the encrypted position values, the subtraction is calculated inside an atomic computation unit. The atomic computations are distributed over all available CPU cores. Afterward, the encrypted subtraction results are collected in a container to be used for verification.

Verification In the verification part, all encrypted proofs are decrypted to get the result of each subtraction. If it satisfies the condition of a position being inside one of the geofences, the geofence label is returned, a `not_found` string otherwise.

Containerization NEXUS-parallel does not need additional dependencies since the parallelization is done via the built-in Python libraries. Hence, the same containerization configuration is utilized, i.e., the same container file is used with the exposed port 4000.

6.1.4 Zero-knowledge proofs

Three ZKP implementations are used in this set membership evaluation scheme. Pequin [237] is used in the previously described evaluation framework that included SHA512 and the homomorphic NEXUS approaches. Besides this scheme, the Zokrates [240] set membership library is tested, and the ZKP Github project [241] by Morais et al. [227] is evaluated. The setups for the latter two are described in Sections 6.1.4.2 and 6.1.4.3. The evaluation results for set membership and range proof of the Morais-based library are described in Section 7.3.5.2.

6.1.4.1 Pequin

Pequin [237] is part of the joined New York University and University of Technology Austin pepper project that aims to provide practical verifiable computation [242] [242]. It is designed as a generic C-compiler for ZKP verifications. At the time of writing, Pequin does not have the complete capabilities of the C-language. The authors confirmed that Pequin cannot utilize *unbounded loops*, *floating-point numbers*, *unbounded recursions*, and *function pointers*. These restrictions affected the implementation process drastically. Inside the Pequin container, the two agents are a Python REST-API that handles the verification requests and a Pequin core module. These two agents communicate with one another to generate proofs.

Input Handling There are two inputs in Pequin, *static* and *dynamic inputs*. Static inputs exist prior to the compilation. Dynamic inputs can be set during the execution. Static inputs are specified in an `inputs` file. Dynamic inputs are passed to the program via a script called `exo0`. This is an executable shell script that echoes input values. This file is known to the compiler and is executed when the program reads the dynamic inputs.

```
1 // Constant values
2 #define AREAS_COUNT 1
3 #define INPUT_COUNT 9
```

Listing 6.5: Pequin Constants

Source Code The program structure is different from the C-language. First, there are constants as shown in Listing 6.5. While `AREAS_COUNT` sets the number of geofences as a static input, `INPUT_COUNT` sets the number of dynamic inputs. As they are hard-coded, the Unix command `sed` is used inside the REST-API to manipulate them. Whenever the REST-API receives geofences or positions, the sed changes the values to satisfy the Pequin requirements.

```
// A list of allowed areas
struct In {
    int64_t data[AREAS_COUNT];
};
```

Listing 6.6: Pequin Input Structure

```
// 1 if private location is in input
    areas, otherwise 0
struct Out {
    int64_t valid;
};
```

Listing 6.7: Pequin Output Structure

Afterward, the *input* and *output structures* are defined, shown in Listings 6.6 and 6.7. An `int64_t` is used as the variable type to represent S2 cells and Geohash values. As the postcode areas have different shapes, meaning different amounts of grid cells, it is impossible to represent these structures using static allocations. The negative value indicates the postcode cells to distinguish them from the geofence IDs. This condition is evaluated in Line 13 of Listing 6.9.

```
// Get user location from external computation
// The script exo0 in bin/ returns a sample location
int64_t* get_current_location() {
    int64_t current_location[INPUT_COUNT];
    // These must be statically allocated to compile,
    // even though we don't need them.
    uint32_t exo0_input1[1] = {1};
    uint32_t *exo0_inputs[1] = {exo0_input1};
    uint32_t exo0_inputs_lengths[1] = {1};
    exo_compute(exo0_inputs, exo0_inputs_lengths, current_location, 0);
    return current_location;
}
```

Listing 6.8: Pequin Input Reader

The last module needed is the `compute` function as shown in Listing 6.9. All computation resides here. While static input and outputs are passed as function arguments, dynamic inputs are fetched via the `get_current_location` function, see Listing 6.8. The `compute` function is a set membership query with two nested loops. When there is a match, the corresponding postcode is returned. Otherwise, it returns 0.

```
// Will produce two lines in the output file
// First line is the return value of the method
// Second line is the value of the output
// Both lines are equal and represent the validity of the location (0 or 1)
int compute(struct In *input, struct Out *output) {
    int64_t* current_location = get_current_location();
    uint16_t i, j;
    int64_t postalcode=0;

    output->valid = 0;

    for (i = 0; i < AREAS_COUNT; ++i) {
        if (input->data[i] < 0){
            postalcode = input->data[i];
        }
        for (j=0; j<INPUT_COUNT; j++){
            if (*(current_location+j) == input->data[i]) {
                output->valid = postalcode;
            }
        }
    }

```

```
23     return output->valid * -1;
24 }
```

Listing 6.9: Pequin compute Function

Setup Pequin After setting up the inputs and constants, the setup script inside the application folder is executed from its parent directory `/pequin/pepper`. This setup file compiles all files and creates the required executables.

Proof Generation and verification There are two additional scripts inside the `/proof_of_location` folder, `prove.sh` and `verify.sh`. These scripts are executed via the REST-API endpoints to generate the proofs and run the verification.

REST-API There are the endpoints, `/geofence_register`, `/generate_proof`, and `/verify_proof`. `/geofence_register` gets the geofence cell values, transforms them to `int64` values, loads them to the Pequin ecosystem, sets the input constants inside the source code by using the `sed` command, and executes the `setup.sh` script in order to compile source code. `generate_proof` gets the position and passes it to the scheme in order to generate a proof via the `prove.sh` script. `verify_proof` makes a call to the `verify.sh` script to get the result.

Containerization As with the previous schemes, Pequin is containerized. In the Dockerfile, the environment variables are set for pre-installation, the required dependencies, and the Pequin and the Python REST-API modules are installed. The REST-API listens on port 7000. For this scheme, the container `debian:stretch` was used as a basis because it is adaptable to Pequin dependencies. Unlike the other schemes, the Pequin container needs more than an hour to install all dependencies, resulting in container caching. Therefore, the changed parts of the Dockerfile are executed for each build. This reduces the build times to a couple of seconds.

6.1.4.2 Zokrates set membership

Listing 6.10 is based on the Zokrates approach by Eberhardt and Tai [240]. The approach is used for privacy-preserving off-chain computations within DLT environments. For the test, an S2 representation of Germany using 95 cells is used (`field[95]`). In the listing, three parent cells (`s2cells[]`) are checked against the geofence.

```
def main(private field[3] s2cells) -> (field):

  field inside = 0

  field[95] gfgermany = [5118182646832168960, 5118270607762391040,
    5118411345250746368, 5118499306180968448, 5118569674925146112,
    5118604859297234944, 5118640043669323776, 5118675228041412608,
    5118833557715812352, 5118921518646034432, 5119097440506478592,
    5119132624878567424, 5148793050549452800, 5148828234921541632,
    5148863419293630464, 5148951380223852544, 5149092117712207872,
    5149232855200563200, 5149373592688918528, 5149461553619140608,
    5149567106735407104, 5156709534269440000, 5156744718641528832,
    5156832679571750912, 5156920640501972992, 5156955824874061824,
    5157026193618239488, 5157096562362417152, 5157800249804193792,
    5157835434176282624, 5158046540408815616, 5158116909152993280,
```

```
    5158152093525082112, 5158240054455304192, 5158380791943659520,
    5158468752873881600, 5158539121618059264, 5158574305990148096,
    5158662266920370176, 5158803004408725504, 5159154848129613824,
    5159506691850502144, 5159594652780724224, 5159700205896990720,
    5159735390269079552, 5159840943385346048, 5159928904315568128,
    5160069641803923456, 5160157602734145536, 5160192787106234368,
    5160263155850412032, 5160298340222500864, 5160403893338767360,
    5160491854268989440, 5160843697989877760, 5161195541710766080,
    5161318687013076992, 5161353871385165824, 5161389055757254656,
    5161477016687476736, 5161617754175832064, 5161969597896720384,
    5162532547850141696, 5163095497803563008, 5163658447756984320,
    5164010291477872640, 5164203805524361216, 5164238989896450048,
    5164344543012716544, 5164432503942938624, 5164573241431293952,
    5164661202361516032, 5164696386733604864, 5164766755477782528,
    5164907492966137856, 5164995453896359936, 5165347297617248256,
    5165910247570669568, 5166262091291557888, 5166350052221779968,
    5166455605338046464, 5166543566268268544, 5166684303756623872,
    5166772264686845952, 5166877817803112448, 5166965778733334528,
    5167599097430933504, 5169006472314486784, 5170290701895729152,
    5170325886267817984, 5170361070639906816, 5172120289244348416,
    5172190657988526080, 5172225842360614912, 5172261026732703744]

  for field i in 0..94 do
    inside = if s2cells[0] == gfgermany[i] then 1 else inside fi
    inside = if s2cells[1] == gfgermany[i] then 1 else inside fi
    inside = if s2cells[2] == gfgermany[i] then 1 else inside fi
    endfor

  return inside
```

Listing 6.10: Zokrates set membership

6.1.4.3 Morais' library

Listing 6.11 is based on the ZKP set membership Github project [241] following the publication of Morais et al. [227]. The test is simplified with a setup of four small integers (11, 42, 61, 71), here called a *map*, as the library does not support more complex structures. In Listing 6.12, a *parent* value 61 is then checked against the set. These are the ZK proof values for integer 61 as described in Section 5.3.6.2.

```
Setup:  {map[11:0xc000092018 42:0xc000092030 61:0xc000092048 71:0xc00025c010] bn256.G2
    ((93213018038109227593644932213403458825306350569167852571871406553047684590 44,
10498459193992345531229748746954530593820660726748174908810965204741822623543),
    (18254291015233316121573675049701663085108620885218234844201414164752899847906,
10288632616163986706142710492890578960281360035453642944921873368951129485500),
    (18250939982366980306876762227368370727314682956897187942360342762948108610873,
20986676706269865442881998603781196631335777904597957077611864475465838878803)) {0
    xc000092008 0xc0003e6040}}
```

Listing 6.11: Set membership test setup of library based on the work by Morais et al.

```
proof_out.s:
    20332816828268944546530457163539975951099387199215341068000944524158216631942
proof_out.V:  bn256.G2
    ((10262548584042236843147522282835667161797841151630654480071129693317899839146,
14858958735085198608438185315246333074959777976420856708168708911108744163285),
    (12152949086026316656180603584610697224088179835142632002491116015001193090906,
19291336388817029526550800694300000157311807200276797346298536680655897898 98),
    (37634086952578287098966279104686607089694410211438249331605572843554384579 13,
```

```
5848138002267333323693360254007145705461143888622787132075638442495789075289))
proof_out.t:
    381963207814214861001200209304223139784332232835541480033202943115763136223 0
proof_out.a:  bn256.GT
    (((134892115968938855930873737269142849241060641662751879838584766313185030389 59,
160614947634863716251305552384877083203693732718517891000442725182488161935 7)
    ,(84146907967602064132995764351381133903108406559171689765008168019044758588 71,
129028971956958939207898510841487594959774360454165801898421935531510009640 39)
    ,(62559017024021315939809131730904774376063667956550132636789940855592723127 75,
114024656594046932222635702968604495764435007971054192176788899611458538214 99))
    ,((14638148011004732362857796067113068769312332742844344442311454048561670974725,
135092397095917144489236291306461722214509653540765204398434801256017864665 19)
    ,(35157371582595754818104777798166064974927207771240912354495674868874540323 69,
788608571344808086197844083980561588347487491868461588835478664894479373700 1)
    ,(95362959518035264533593084046540412688963771617447212247825022944894840065 88,
170105353223441761460379215916973281688523109885614508365226597474938053414 88)))
proof_out.D:  bn256.G2
    ((17769373058677899910430804176598888879478211338155041869030432087017071816 52,
163590702666396334031151526012392301912579160595915198122060334165085683296 53),
    (51366928365139448024421150814733814525522054178695418566736168338704568760 33,
857559758416680952249667752904495029998873556675467933077196001211333887931 7),
    (85640330915010964955884237986966757812782933030362209041503783626257114218 46,
771224337857833737232064591746677608091283307193542507461376659324242234595 1))
proof_out.C:  bn256.G2
    ((63421018638915329941229525318132140486381058759067601670738803771219014445 79,
138585145116101053749116404480932375917400331930763513742836820749085118418 27),
    (81760589942957760474601820731381539553657706947372685854285643740022217406 73,
286039599822641381048027268110758467267482420060928677260361236851217049027 7),
    (35120309880666367319310945801856816813894181257071259155317411410085013068 15,
149976205904223316397490739659558425568741249222959667600572028210916321051 23))
proof_out.c:
    107795592788023501662805379017865119247202695323053051204862280466283649369 87
proof_out.zr:
    960129215251026417684297743707331781566620789613244751511849906953476551896 3
proof_out.zsig:
    194269869765038410708098175122810011996138777410727590292871592771022103442 45
proof_out.zv:
    171156367613482958741086558813727835246163346578761543774564400846032762904 09
```

Listing 6.12: Set membership proof for integer 61 in map.

6.2 Attribute-based encryption

Three different components have been implemented during the ABE-based research, the *JCPABE2 library* [243], a server component named *attribute authority (AA)*, and a *mobile application* based on the Android platform, previously published by Denisow et al. [197] and Zickau et al. [190].

6.2.1 JCPABE2

The library implements the five basic functions plus additional functionalities described in the next section. To implement this library, it has been decided to build upon the Java CPABE version [244]. The most compelling argument for Java CPABE is its portability. Since every component has been written in pure Java 6, it can efficiently be run on the Android operating system. The implementation is called JCPABE2 [243]. The following

paragraphs will overview the introduced additions and changes made to the original Java CPABE library.

Policy Parser The C-implementation that had been ported uses two different policy representations. All Boolean formulas are transformed into the internal format. Then this format, which is also human-readable, is used to encrypt the file. The Java port of the CPABE implementation lacks this transformation. This is why the parsing capabilities of the C implementation in Java were re-implemented. It is now possible to traverse the parse tree and generate the internal policy used by the ABE part.

Attribute Parser Adding more complex attributes, such as numbers and locations, requires parsing these attributes when generating private keys. This is simpler than parsing the policy since the attributes have neither operations nor a specific order. Because of this simplicity, it was possible to implement the parsing with the help of regular expressions.

Hashing of Attributes The origin implementation saved the name of the attribute in plaintext. This is a problem because the name of an attribute may already disclose information to an attacker, e.g., an attribute called *berlin*. It would be easy to determine that this attribute is a location attribute by analyzing the policy tree. With plaintext attributes, it would also be possible to use the internal representation of the location attribute to determine which Geohash has been used to create these attributes in the first place. With this information, an attacker may either go to that location or even try to spoof the location to access a file that he should not access ultimately. All names of attributes in private keys and the policy formulas of the encrypted files are saved after they have been hashed.

6.2.2 Attribute Authority

The AA is an implementation of a key authority. It manages the public keys and the secret master keys. It is called AA because it is a web application that can distribute generated attributes. There are two separate parts to the AA. There is a REST-API and a web application that serves as a *management tool (MT)* (Figure 6.2). Both use the same SQLite database to retrieve information.

Username	Password	Change PW	Delete
owner	New Password	Change Password	Delete
normalUser	New Password	Change Password	Delete
testUser	New Password	Change Password	Delete
Name	Password		Create

Figure 6.2: Screen capture of the user management in the management tool.[2]

REST-API The REST-API is used for communication with the client application. Its primary function is to distribute generated attributes. It also has an index for the secret

[2]Image sources: Screenshots in this section are of the ABE web and mobile app interfaces implemented by

and public keys. There are two types of users in this system *authenticated* and *unauthenticated*. Authenticated users are associated with a username. An overview of the REST-API methods is as follows. Unauthenticated users may do the following.

- `GET /connectiontest` It can be used to test if an attribute authority is reachable.
- `GET /publickeys` Lists all available public keys.
- `GET /publickeys/<name of a key pair>` Retrieves the public key that has the given name.
- `GET /publickeys/<name of a key pair>/attributes` Returns a list of attributes that can be generated for this key pair. Each element in the returned list contains information about the name of the attribute and the type of the attribute.
- `POST /publickeys/<name of a key pair>/attributes` Requires the user to send a list of the requested attributes and the proposed values of them. It also requires a value from the private key of the user. Returns the generated attributes.

Authenticated users can additionally use the following methods.

- `GET /authtest` It can be used to test credentials.
- `GET /secretkeys` Retrieves a list of all secret master keys of which the authenticated user is the owner.
- `GET /secretkeys/<name of a key pair>` Retrieves the secret master key of that name. Requires that the requestor is the owner of the key.
- `POST /secretkeys/<name of a key pair` Creates a public/secret master key pair with the given name. The user requesting this operation is marked as the owner of that pair.
- `DELETE /secretkeys/<name of a key pair>` Deletes the public/secret master key pair and all associated data. Requires that the requestor is the owner of the key.
- `POST /privatekeys/<name of a key pair>` Requires a list of attributes. Returns a newly created private key for this key pair. Requires that the requestor is the owner of the key.

Designation	Owner Name	Action
aaa	owner	Delete
aab	owner	Delete
aad	owner	Delete
praxisTest	owner	Delete
aac	normalUser	Delete
new designation	Choose User	Create

Figure 6.3: Screen capture of the key management in the management tool.

Management tool The MT is used to control everything related to the RESTful service. It is not meant to be used by an end-user. The MT features user and key management (Figures 6.2 and 6.3). Users, as well as master keys and their associated public keys,

Iwailo Denisow in the context of the Curcuma research project.

can be created and deleted. It can be used to change passwords. The most important feature of attribute generation is the management of attributes for each key pair. This controls which attributes users can generate through the REST-API of the AA. The name and the type of the attribute are needed for creation. Currently, four types of attributes are possible: *static string*, *string*, *number*, and *location*.

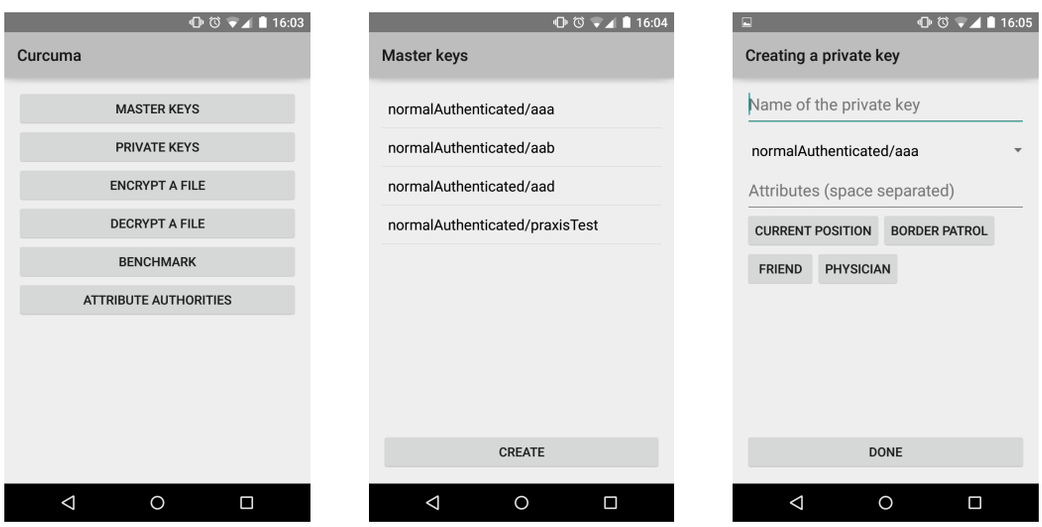
User friendly Attribute Name	Attribute Type		
currentLocation	location	Change	Delete
dynString	string	Change	Delete
someNumber	number	Change	Delete
staticString	staticstring	Change	Delete
attribute name	Choose Type		Create

Figure 6.4: Screen capture of the attribute management of a key pair in the management tool

6.2.3 Android Application

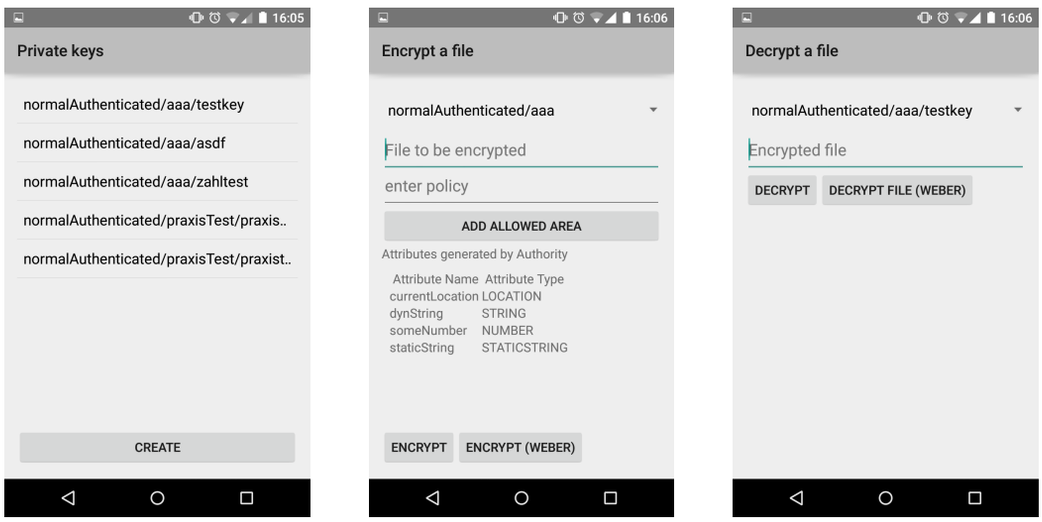
A proof of concept mobile client application on the Android platform has been developed. Since the application is tightly integrated into the architecture, nearly every action communicates with an attribute authority. Screenshots of the different interfaces are given in Figures 6.5 to 6.7. There are five entries in the main menu of the app.

- **Attribute Authority**: Lists all registered AAs. It is possible to add or delete AAs. Adding an AA requires a name and a URL.
- **Master Keys**: Displays owned secret master keys stored on AA level. Create and delete secret master keys.
- **Private Keys**: Shows all locally stored private keys. Create (with adding ad-hoc or pre-defined attributes) and delete private keys.
- **Encrypt a File**: Set a public key, choose a file, and create a policy (including area selection on a map); the file is encrypted and stored locally.
- **Decrypt a File**: Set a private key and a file. Requests new attributes from the AA.



(a) Main menue (b) Master keys (c) Private key creation

Figure 6.5: Android ABE app screenshots



(a) Private keys (b) Encryption (c) Decryption

Figure 6.6: Android ABE app screenshots

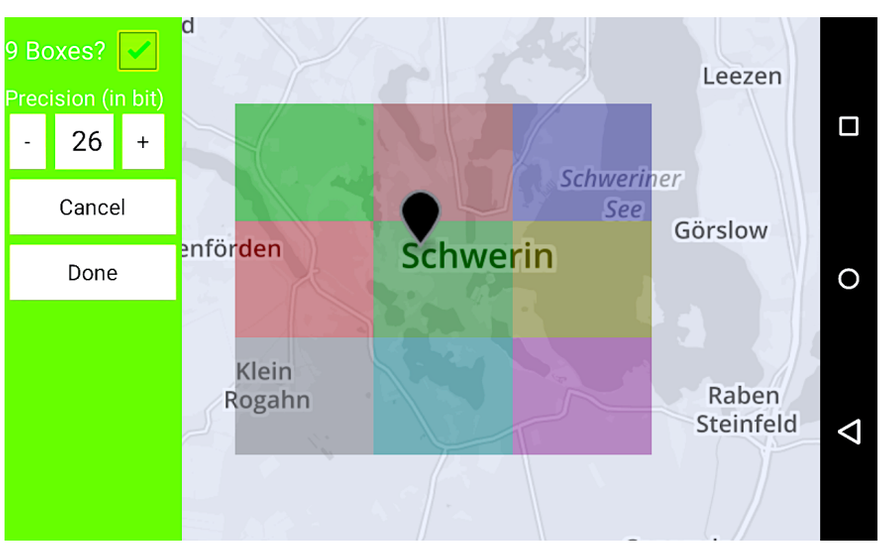


Figure 6.7: Android ABE app screenshot, map options

6.3 Policy-based scheme

As it can be argued that the ABE scheme also includes a policy-based approach to geofencing, ABE relies heavily on cryptographic primitives. The following results show the whereabouts use case interfaces within the PARADISE privacy project, independent from cryptographic schemes. In the context of the anti-doping scenario, we have doping control officers who need to locate athletes to carry out unannounced urine or blood tests. An application has been developed that prototypically implements the conceptual privacy geofences, also referred to as *privacy gardens*. This application is shown from the perspective of a NADO operator, an athlete, and a DCO.

6.3.1 Operator view

The NADO operator and athletes are given an interface to define general and personal privacy geofences with input tools similar to other map-based interfaces, consisting of different shape tools, such as circles and polygons (top left and right corners of Figures 6.8 to 6.10).

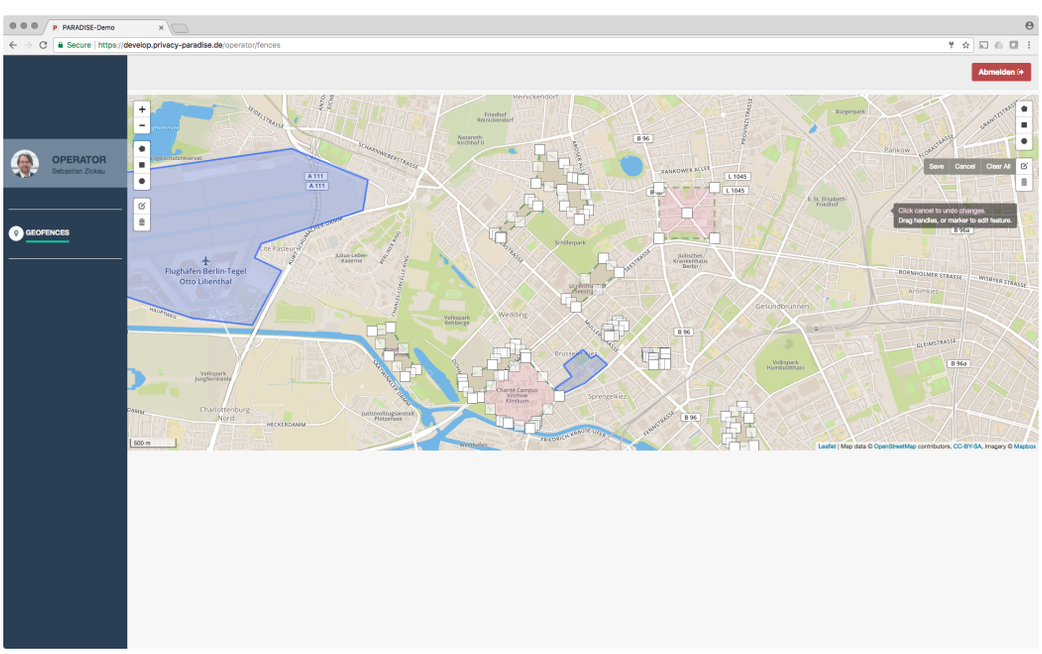


Figure 6.8: NADO Operator interface for creating general privacy areas (green).[3]

A NADO operator is given the task of defining general privacy geofences (green, Figure 6.8) and areas in which an athlete is not allowed to define privacy areas (blue, 6.9).

6.3.2 Athlete view

In Figure 6.10, a screenshot of the application exemplifies the view and notion of different geofences for the role of the athlete. In the figure, the blue geofences are banned areas. The red geofences are personal private ones but not accepted by the system as they are

[3]Image sources: All PARADISE and Eden web interfaces use the Leaflet open-source JavaScript library. The library is developed by Vladimir Agafonkin, now employed by Mapbox. The underlying map data is provided by OpenStreetMap (OSM) under the Open Database License (ODbL), openstreetmap.org/copyright.

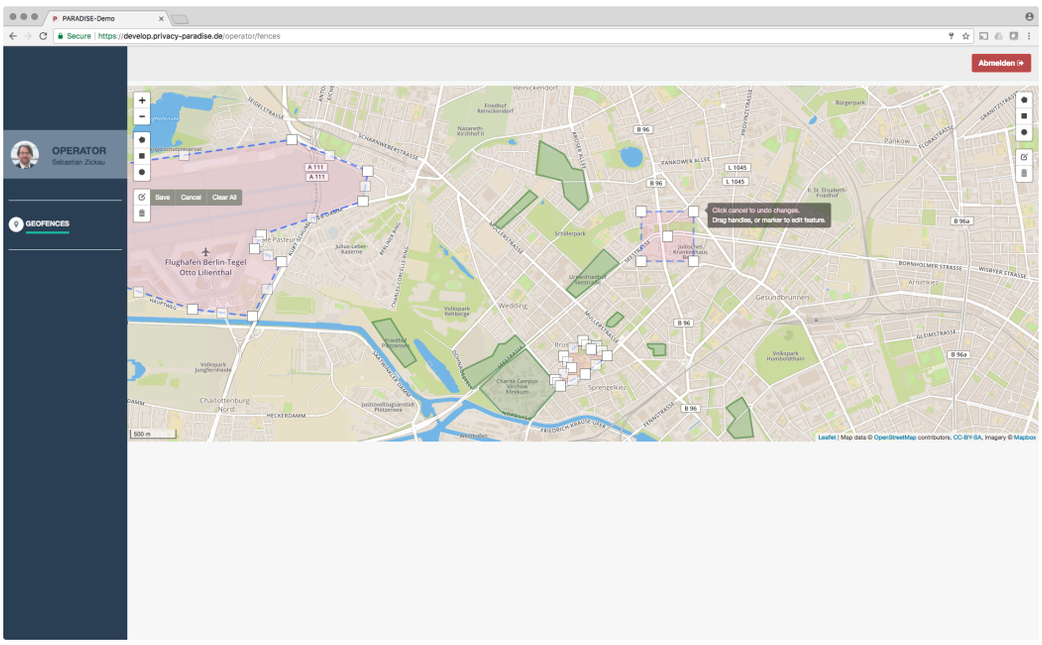


Figure 6.9: NADO Operator interface for defining areas in which personal privacy geofences are not allowed (blue).

either too large or overlap a banned geofence. The green areas represent general private geofences. Athletes can deactivate the latter as they may overlap a place where an athlete can or should be located, e.g., a campus or a workplace.

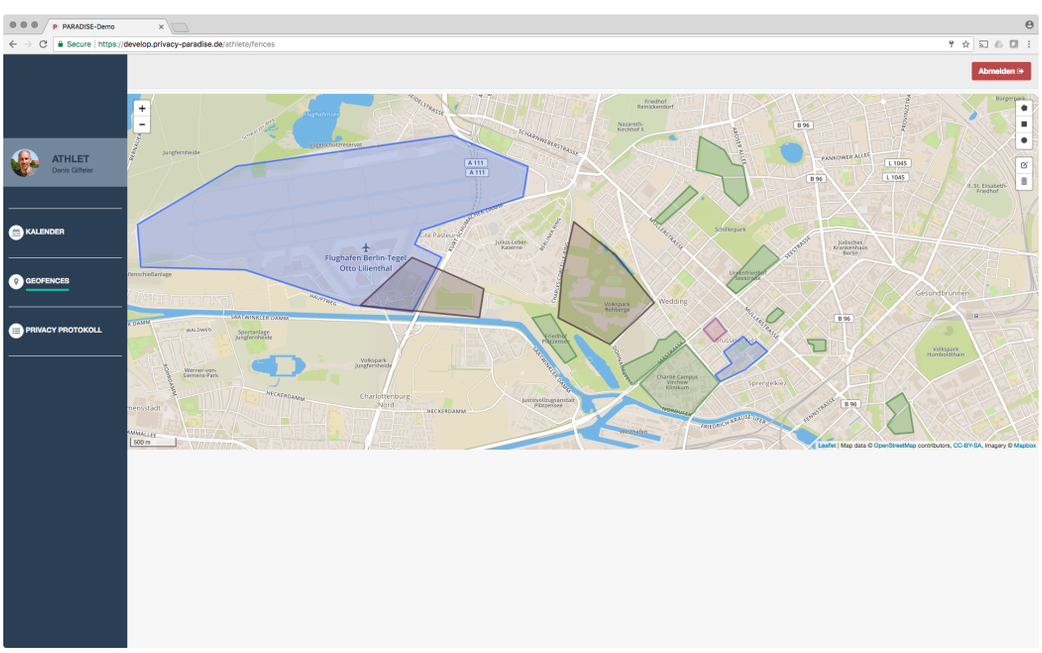


Figure 6.10: Athlete interface for creating personal privacy areas. The example shows arbitrary-shaped geofences, for example, around two cemeteries in Berlin (green). It also shows two red geofences, which are areas not excepted by the system as they are either too large or overlapping blue areas. These indicate areas an athlete is not allowed to declare private. An airport and a sports field are chosen as examples in the screenshot.

6.3.3 DCO view

From a DCO perspective, the application shows different results. A DCO gets to know information from a calendar view of an athlete with imprecise location information, such as city or state levels. This is illustrated in the screenshot of Figure 6.11.

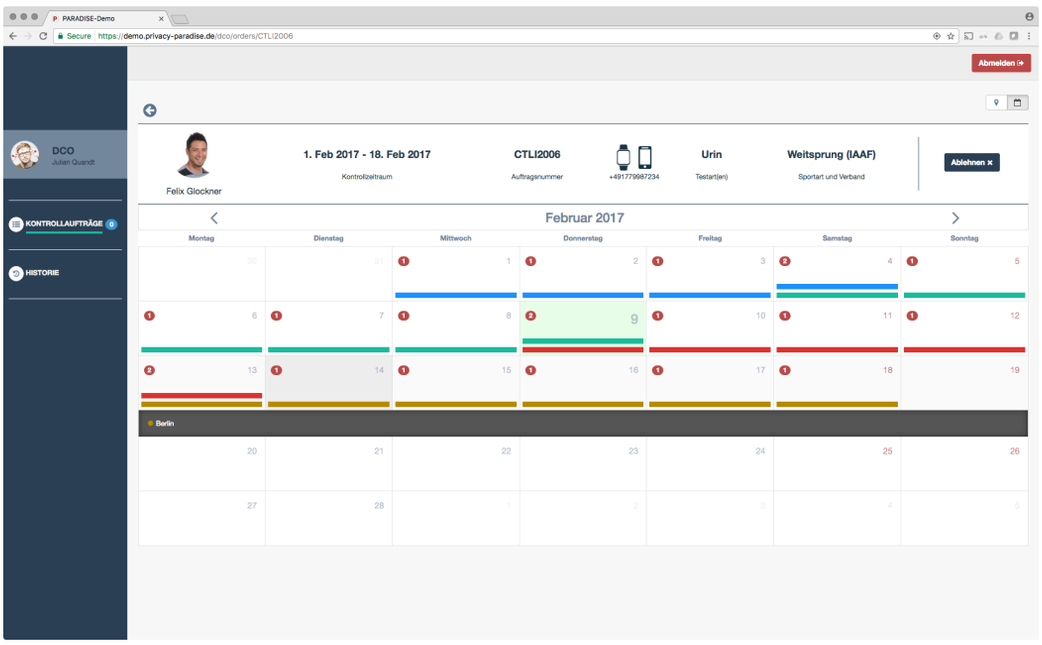


Figure 6.11: DCO view of an athlete simplified calendar including imprecise location information at city level, in this case, Berlin (green). The other colors imply that the athlete is in different cities or areas on the respective dates.

The DCO view on-the-go is different, though. Depending on the position of the DCO relative to the athlete, he needs to find and rely on the athlete's position with respect to general and personal privacy geofences. Figure 6.12 shows a DCO position in Berlin and provides an area in which the athlete is located due to the distance between the athlete's actual position and the DCO. At this point, the DCO does not need to have accurate whereabouts of the athlete as the meeting between the two actors is not likely to happen imminently. In the figure, only the postcode area is returned to the device of the DCO. The color green indicates that the athlete is *not* within a personal or general privacy geofence.

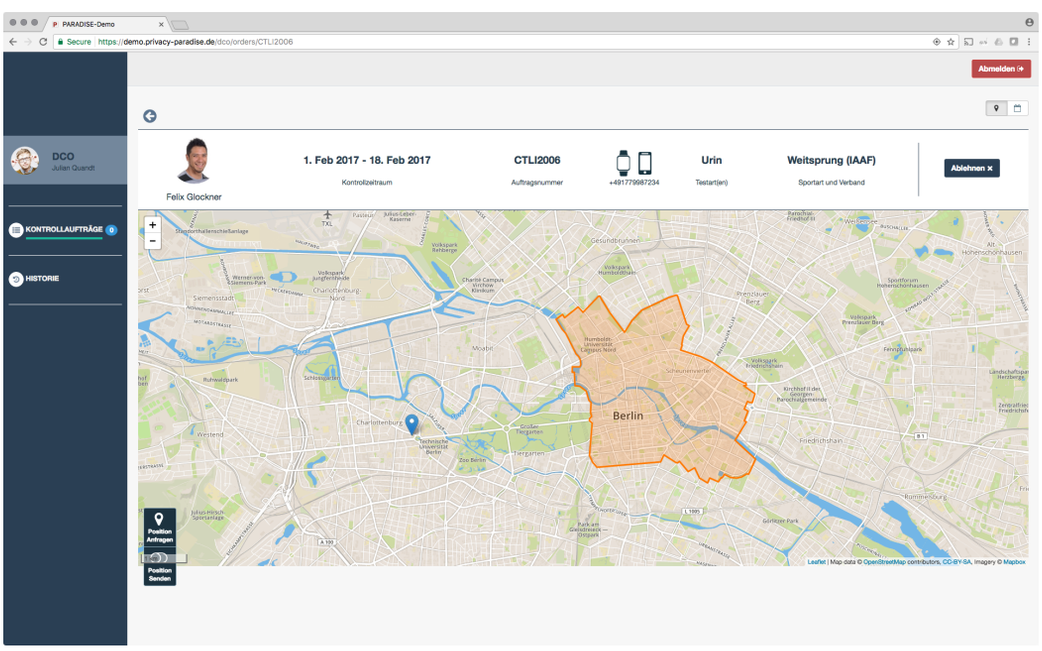


Figure 6.12: DCO view of obfuscated whereabouts, i.e., a Berlin postcode area, of an athlete (green).

Unlike Figure 6.13, here, the DCO is close to the athlete, indicated by the DCO marker within the geofence. The red color and the text indicate that the athlete is within a privacy area, either general or personal. This differentiation is not reflected in the DCO's view. This situation would demand calling the athlete to arrange a meeting place or another reactive location request to check if the athlete has left the private geofence.

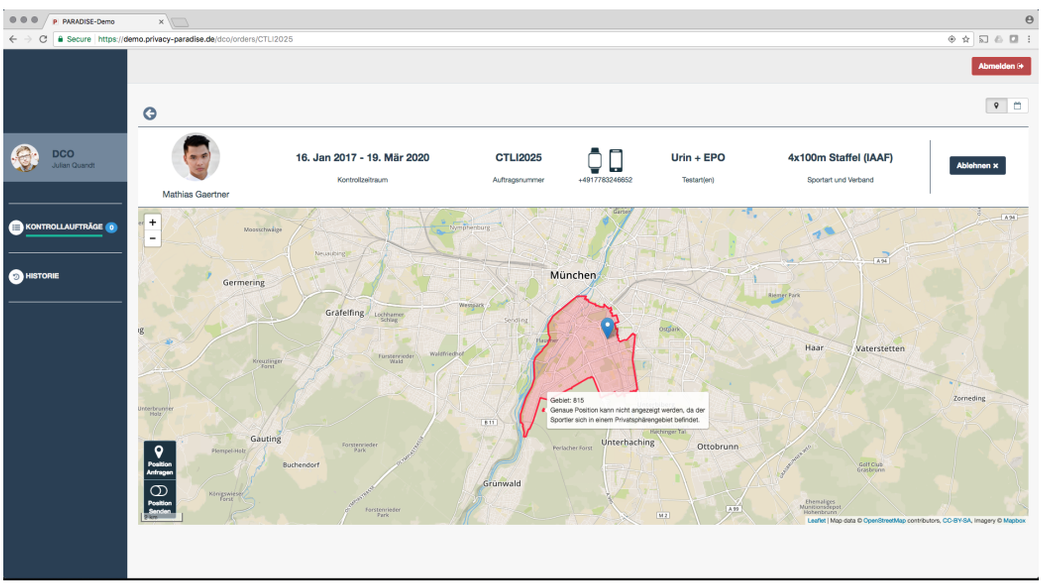


Figure 6.13: DCO view of the location of an athlete who is inside a privacy area, the overlying geofence (red) for the German five-digit postcodes starting with 815.

The project discussed what kind of limitation a presence within a private geofence could be implemented, e.g., a time-based approach could have been implemented to reveal the location after several hours. However, this could indicate why an area is considered private.

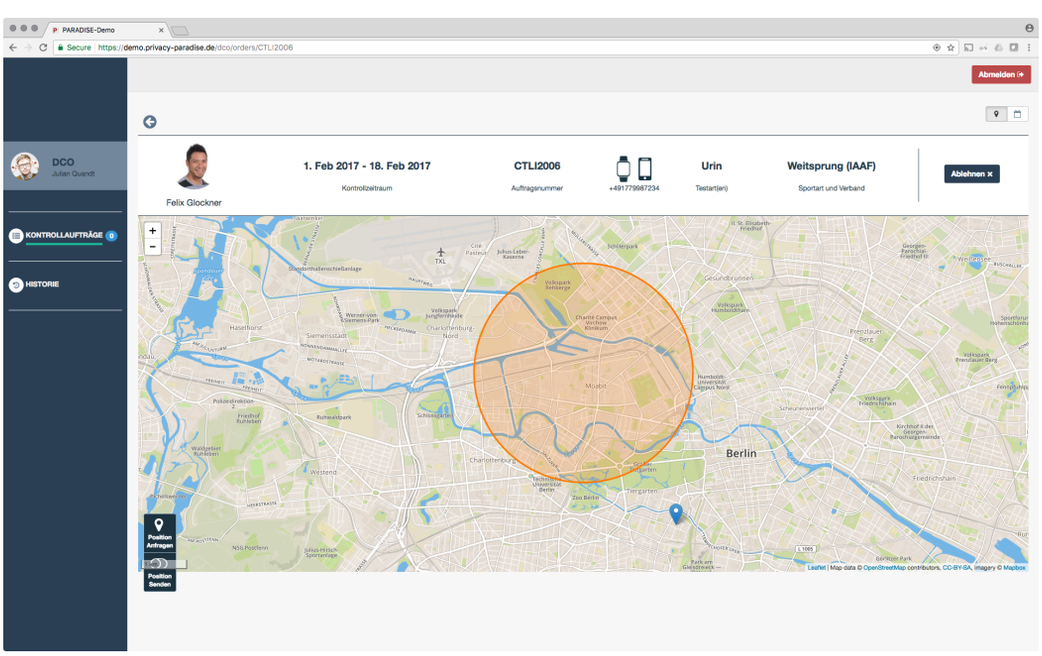


Figure 6.14: DCO view of athlete whereabouts details with no privacy implications (green). The DCO is far away, no further details are presented.

Figures 6.15 to 6.17 show the case in which a DCO is located close or very close to an athlete. A meeting is likely to happen in the imminent future. Therefore, the where-

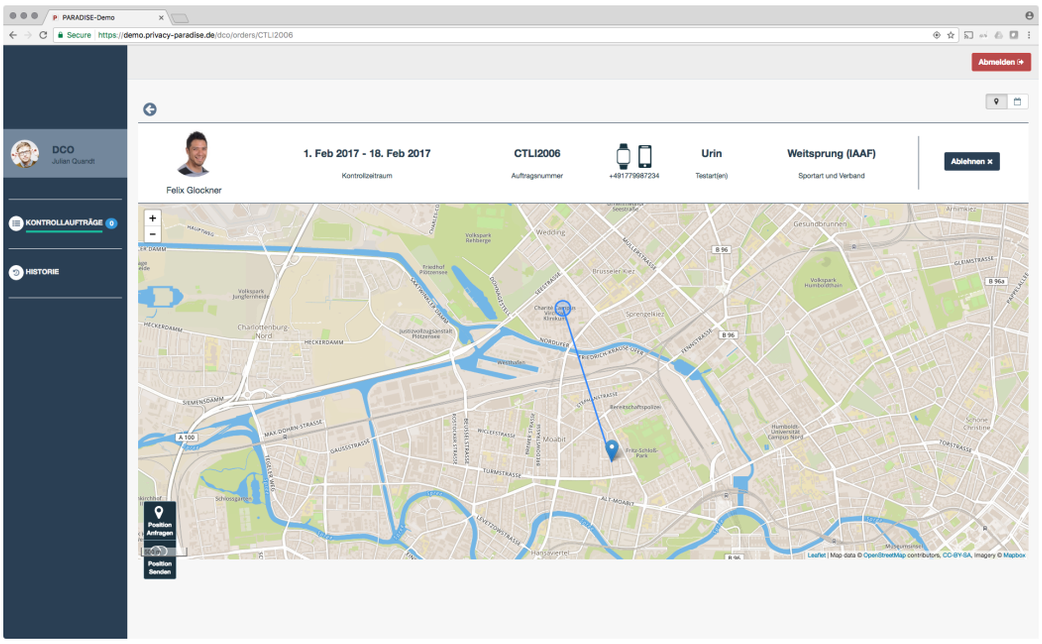


Figure 6.15: DCO view of athlete whereabouts details with no privacy implications. The DCO is getting closer, the athlete area is getting smaller (blue).

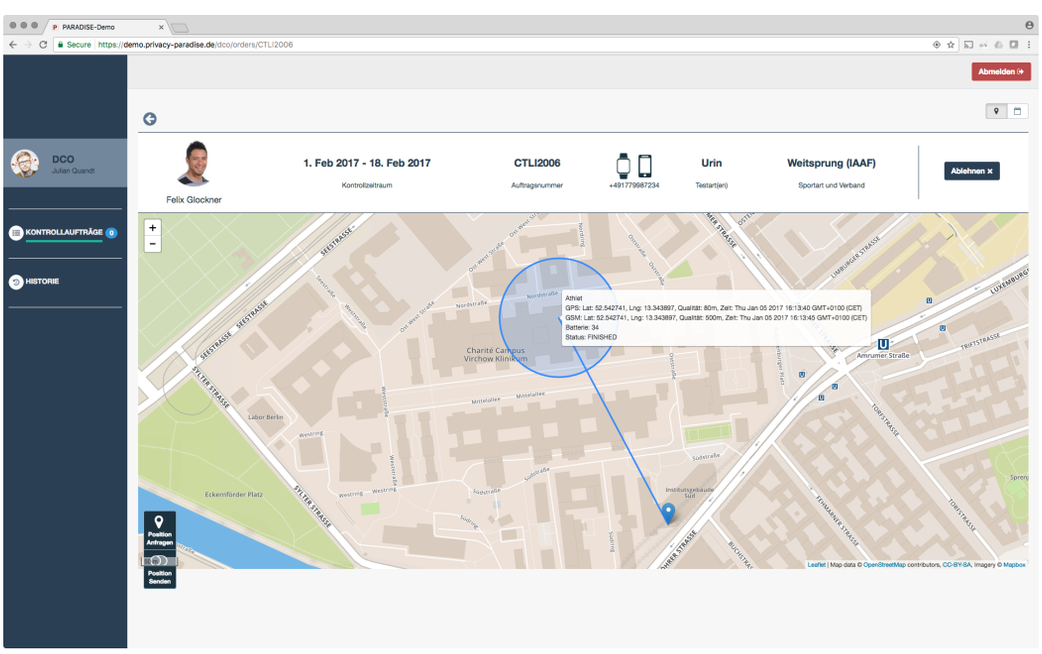


Figure 6.16: DCO view of athlete location details without privacy implications. The DCO is nearby, the athlete location data is more detailed (blue).

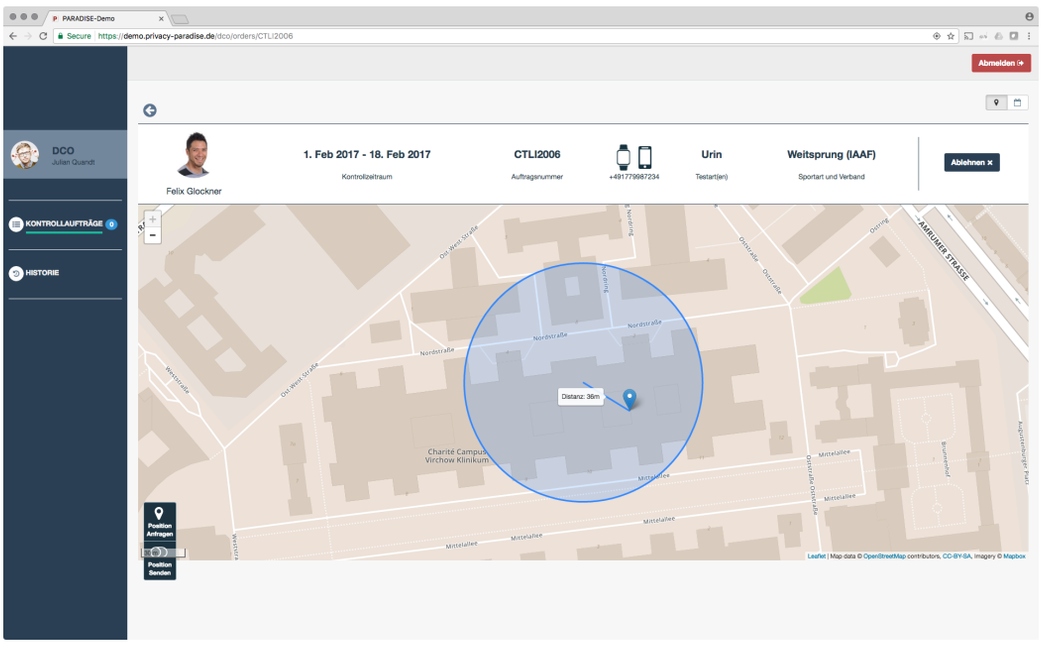


Figure 6.17: DCO view of athlete whereabouts details with no privacy implications. The DCO is very close, the athlete area is just a small circle (blue).

abouts information is progressively disclosed to the DCO, including latitude and longitude with accuracy value, date, time, timezone, and battery power of the positioning device, i.e., the wearable. It has to be emphasized again that these values were shown in the implementation of the project prototype.

6.3.4 Semantic hierarchies and general privacy areas

During the realization of components for the whereabouts use case, the previously described concept of general privacy areas was formulated. To make it easier for the operators to define such areas, queries could be specified that gathered geofence information. The description of the demonstrator and findings in this section were partly previously published in [38]. The example service supporting PARADISE is called *eden*.

To get semantic information in addition to geofence structures for specific areas within cities, the Nominatim service that OpenStreetMap provides was chosen [245]. It can be queried for geocoding and reverse geocoding information. Unfortunately, the quality of returned results depends on the investigated area. For popular areas, such as Berlin, the data was useful. For other, more rural areas, it was not. The test data chosen for the demonstrator is for the city of Berlin, see Figures 6.18 and 6.19.

Getting geofence information from institutions, such as university campuses, was different in quality. That led to the decision that the last area in the geofence hierarchy was always presented to the DCO as a circle with a predefined radius.

On the server-side, the REST interface of the privacy area concept is realized using the Django REST framework [246]. The lightweight NGINX HTTP server and the Green Unicorn load balancer [247] handle the requests. Google's Cloud Messaging Service Firebase [248] was chosen for the communication between the Android OS and the backend. The PostgreSQL database is included for address and athlete details and its spatial PostGIS extension for evaluating distances and geofences. The database object

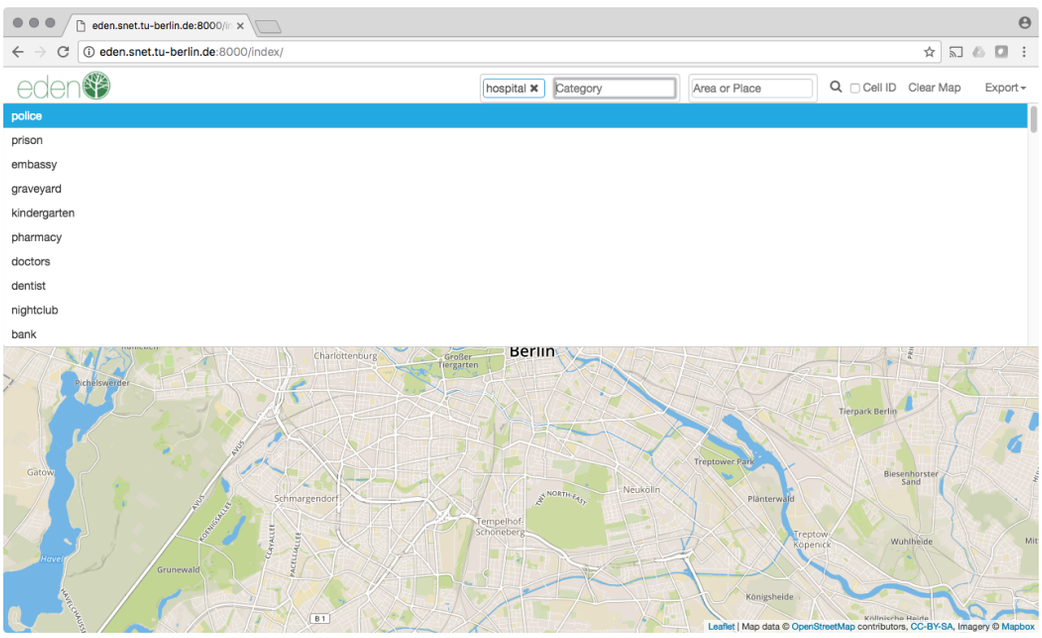


Figure 6.18: Geofence interface for querying specific categories for an area. The information is publicly available through the Nominatim service. Here the category 'hospital' is chosen.

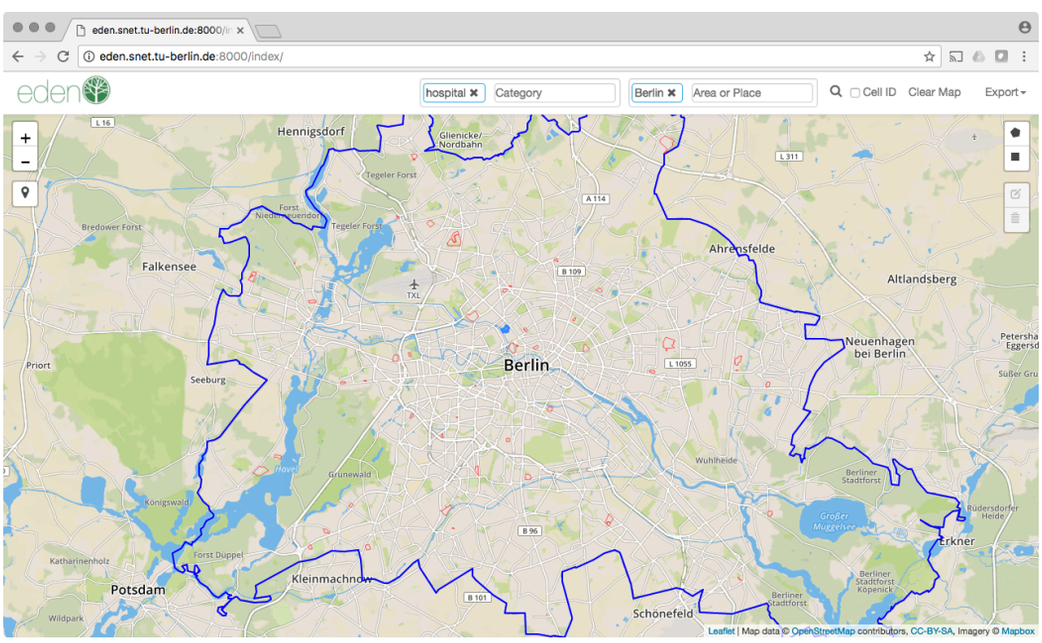
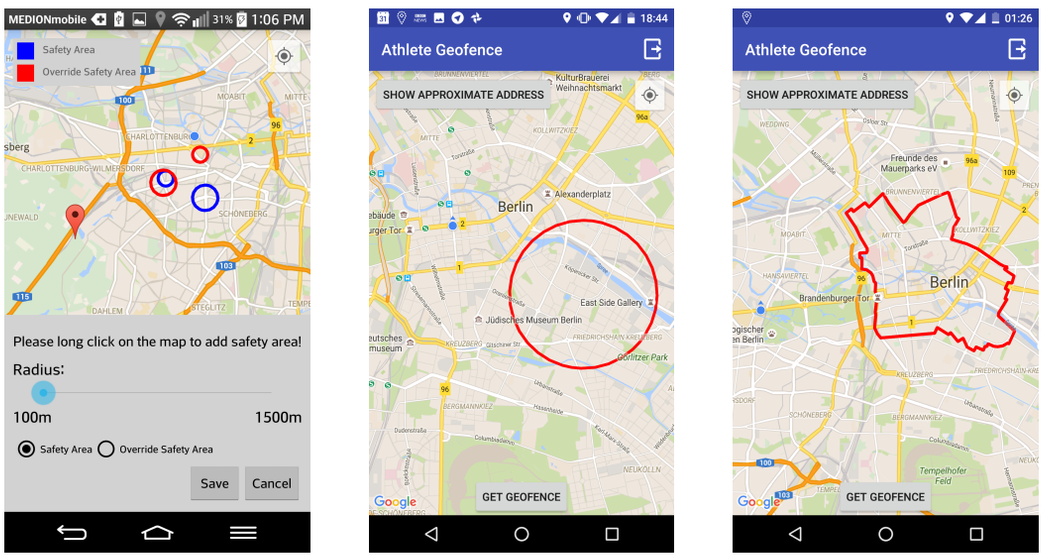


Figure 6.19: Geofence interface for querying an area for specific categories. The information is publicly available through the Nominatim service. Here the area 'Berlin' is chosen.

graph consists of eight entities: location, address, event, user (authentication information included), athlete (Firebase-ID included), DCO, privacy area, and geofence. A Google service is integrated into the Android application for calendar integration. To represent the geofences, the widely used GeoJSON [249] format was chosen. Time constraints are checked against the current time. If the app is started for the first time, a Firebase token is generated for the device user. The Android interfaces are shown in the screenshots of Figure 6.20.



(a) Athlete interface (b) Simple geofence (c) Presented geofence

Figure 6.20: Screenshots of the Android demonstrator.[4]

6.4 Distributed ledger technologies

As visualized in Figure 7.46, the provided position may consist of multiple grid cells. To determine whether the oracle-provided position overlaps with the geofence, each grid cell needs to be checked, potentially with multiple reduced cell levels. An overlap exists as long as one cell version is contained in the geofence. It would also be feasible to enforce full containment by requiring every cell to find a match in the geofence. The corresponding algorithm is shown in Listing 6.13.

Examining the algorithm, we observe that the number of cells used in the geofence has no impact on the number of key-value lookups that need to be performed. If c is the number of grid cells to be checked, and l is the number of levels the grid cell needs to be reduced by at most, then an overlap can be determined in at most $c \cdot l$ lookups. Therefore, the lookup depends on the number of different resolution levels used in the geofence and how many cells are supplied by the location oracle. Parts of the geofence research carried out in distributed ledgers was previously published by Victor and Zickau [121].

```
1 hasOverlap(locationList)
2 Predefined: geofenceMap, minLevel of geofenceMap
3 Input:      Key-value map of the geofence, list of geolocations to be checked
```

[4]Image sources: Underlying map data from Google Maps 2018 GeoBasis-DE/BKG (2009), Google, created and used via the Android API in the mobile demonstrator of the project.

```
Output:     Boolean indicating whether the locationList is at least partially
            inside the geofence

for cell in locationList do
    while level(cell) <= minLevel do
        if cell in geofenceMap then
            return True
        else
            cell = reduceLevel(cell)
        end
    end
end
return False
```

Listing 6.13: DLT geofence algorithm of smart contract.

6.5 Chapter summary

The chapter mainly presented different parts of the proof of concept implementations. The evaluation framework was specified, which built the foundation for comparing the hashing scheme, the homomorphic encryption approach, and the zero-knowledge proofs. Additionally, some of the tested ABE libraries and interfaces were described. The policy-based interfaces for the whereabouts use cases were shown. Finally, the DLT smart contract algorithm was presented.

Keep your friends close, but your enemies closer.
The Godfather II (1974)

7 Evaluation

This evaluation chapter is divided into the following parts. First, the evaluation processes of the design science methodology are addressed. Second, an overview of the evaluation of privacy methods is provided, including how they address the 17 privacy objectives and an assessment of which method is best suited for the LBS classes following the definitions in Chapter 3. Third, a large section of various computational evaluation results is given. Starting with a geofence representation evaluation, showing why the S2 representation is used for most approaches. The computational comparisons include *grid cell hashing*, *probabilistic methods*, *homomorphic encryption*, *zero-knowledge proofs*, and *attribute-based encryption*. The fourth section overviews the results of the whereabouts use case. The fifth section presents findings of the geofence use in distributed ledger environments. The chapter concludes with a summary.

7.1 Methodology

In Table 7.1, the evaluation methods, together with examples of the design science process, are shown. The process is described in Chapter 2. Based on the related use cases described from the list of objectives, the approaches and methods described in Chapter 5 are evaluated. All five design science methods [64] are used during the evaluation, referenced by the section numbers in the third column.

7.2 Addressing the privacy objectives

The proposed solutions will meet the following privacy objectives.

7.2.1 Objectives

Table 7.2 shows the results of the approaches presented and how they address the objectives derived from the use cases in Chapter 2. In the motivation chapter, Table 2.2

Table 7.1: Design evaluation methods

Method	Description and examples	Sections
1. Observational	Case Study: Study artifact in depth in business environment Field Study: Monitor use of artifact in multiple projects	7.2, 7.3.7, 7.4, 7.5
2. Analytical	Static Analysis: Examine structure of artifact for static qualities (e.g., complexity) Architecture Analysis: Study fit of artifact into technical IS architecture Optimization: Demonstrate inherent optimal properties of an artifact or provide optimality bounds on artifact behavior Dynamic Analysis: Study artifact in use for dynamic qualities (e.g., performance)	7.2,7.3, 7.3.1,7.3.2, 7.3.3, 7.3.4, 7.3.5,7.3.7, 7.5
3. Experimental	Controlled Experiment: Study artifact in controlled environment for qualities (e.g., usability) Simulation and Execute artifact with artificial data	7.3, 7.3.2, 7.3.3, 7.3.4, 7.3.5, 7.3.7, 7.4, 7.5
4. Testing	Functional (Black Box) Testing: Execute artifact interfaces to discover failures and identify defects Structural (White Box) Testing: Perform coverage testing of some metric (e.g., execution paths) in the artifact implementation	7.4, 7.5
5. Descriptive	Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifacts utility Scenarios: Construct detailed scenarios around the artifact to demonstrate its utility	7.2, 7.4, 7.5

shows how the use cases generalize the 17 objectives. The following subsections provide an overview of how the ten approaches differ in their application *with respect* to the objectives.

Objective 1: Data send to service The thesis primarily addresses use cases involving only a target client and a location-based service. The goal was to make trusted third parties obsolete. This is reflected in the location data sent from the target to the service in all approaches. As mentioned in the related work chapter, homomorphic encryption usually requires the involvement of a TTP that evaluates the encrypted data against the predefined geofences. An additional party is required in the presented HE case, but this could also be avoided with the *HE-XOR-ZKP* approach.

Objective 2: Precise target position The target position and the accuracy of the geofence should be as accurate as possible to allow for a variety of use cases that are also not constrained by such a factor. The table shows that all ten approaches support this objective naturally by using the grid cell geo encodings described in Chapters 3 and 5.

Objective 3: Fast encoding In some use cases, location updates must be performed regularly with short time intervals, e.g., for fast-moving targets. That requires fast encoding of the current location and the application of necessary cryptographic primitives within the approach. The table shows that methods based on hashing and generic mathematical operations are well-suited for such an objective. Other techniques based on more complex computations, such as HE, MPC, and ZKP, are less suitable. The evaluation chapter also includes dedicated sections on performance comparisons for GCH, HE, ZKP, and ABE schemes.

Objective 4: Service learns what's necessary This can be considered the most critical objective. The service should only learn what is necessary to perform its function. The grid cell hashing approach can reveal derived information over time, as mentioned in Section 5.3.1. Also, the policy-based approach of the whereabouts use cases makes a difference in what the service learns and what its user, the DCO, learns about the target location. Therefore, the combination of, for example, hashing and policy-based approaches are recommended. The table shows that schemes solely based on hashing algorithms risk revealing additional information.

Objective 5: Geofence not known to service This objective differs depending on the underlying use case. In some situations, the service and the target must know the geofence, such as carsharing applications. Alternatively, it may be that the target hides the location of the geofence and its semantic reason from the service (whereabouts). This objective is handled differently in the approaches. HE, CA, and ABE schemes support both, indicated by the semi-circle in the table. This can also mean a combination of primitives.

Objective 6: Service may limit geofences Following the last objective, the service may not know the geofence(s) of the target but may want to limit their size or number. This is especially important for the whereabouts use case since the service does not want the athlete to declare a large area as private. Only PPM and policy-based approaches fit

this objective. PPM can be combined with other approaches as the first step of the setup phase to ensure limitations. Due to their potential computational versatility, HE, MPC, and ZKP can also be designed to suit this objective.

Objective 7: Privacy-preserving distance measuring The grid cell encoding also enables privacy-preserving distance measurement between two clients. The hashing approach is well suited for comparing client-encoded grid cell levels when both use the same salt values. Exchanging salt values while two users are in close proximity hides information from services. As the services are not present at all the relevant locations, they cannot get to know secret information. The semi-circles indicate the schemes that can be adjusted to address this objective.

Objective 8: Position shown if outside The whereabouts use case requires the DCO to determine the exact position when she is in the vicinity of an athlete who is not in a private area. In addition to evaluating a position against a geofence, ABE can also encrypt and decrypt a payload in the ciphertext. Using, for example, the inverse geofence approach enables this objective. The policy-based approach was also designed to allow for this feature but is likely to reveal additional information to the service but not to the DCO.

Objective 9: Arbitrarily shaped geofence The S2 and Geohash grid cell encoding at the centimeter-level allows the use of arbitrary shapes, such as postcode areas and road networks. All approaches support this objective.

Objective 10: Feasible memory sizes As memory is cheaper than RAM or computing power, this does not limit the approaches. It depends on the number of geofences, their accuracy, and the number of target users. In this chapter, memory sizes are compared for different schemes.

Objective 11: Geofence updates not necessary (privacy) As exemplified with the grid cell hashing approach, a derivation of knowledge about a target client can be observed over time, see Objective 4. A regular instantiation of the encoded cell values is necessary. The correlation between client values is not derivable for the service for the other approaches.

Objective 12: Low-powered device The power required to generate the encoded or encrypted client position is most notable for HE, MPC, PSI, and ZKP schemes.

Objective 13: Geofences may be added, deleted, or updated Over time, the target or service may want to add, delete, or update geofences. This works well for HE, CA, ABE, and policy-based approaches. CA variants, such as the dynamic accumulator schemes, already include updates. In the ABE scheme, the generation of new private keys can include new grid cells. GCH, MPC, PSI, ZKP, and PPM require either updates of initial setups or additional computations. General Bloom filters are less suited for updates, as deleting values is not a feature of the original version, but new ones, such as counting Bloom filters, can be adapted.

Objective 14: On-the-fly geofences This objective is similar to the previous one. However, on-the-fly updates should enable the traffic congestion use case for carsharing, i.e., updating the geofence should not cause a service to narrow down the area to know the car's location. CA and policy-based approaches are most suited. A TTP with HE also enables this particular objective. MPC, PSI, ZKP, and PPM can be adapted to achieve this objective.

Objective 15: Separation of positioning and encoding In most approaches, the temporal uncoupling of the target location and the encoding for a privacy evaluation can be separated. Therefore, the PSI schemes require back-and-force communication and are unsuitable for this objective. The objective mentioned in Chapter 2 was based on the cycle path evaluation. All methods can be used because this particular use case does not require on-the-fly computations.

Objective 16: Services cannot get shared user secrets In the case of secure salt values for privacy-preserving distance measurements, users share these secrets. A service should not be able to *disguise* as a user to "*steal*" such a secret. In general, the acquisition of such secrets should be made difficult, for example, by sharing secrets directly between users or by limiting the number of users who share secrets among them. Also, the target should have control over the generation of private keys that a service stores. Schemes based on public and secret keys, HE and ABE, and schemes based on aggregated data, probabilistic methods, and CAs, are most suitable for this objective.

Objective 17: Support BLE, Wi-Fi, and other identifiers The grid cell approach uses private evaluation of sets to determine if a client is within a specific area. The position and the geofences are set elements, either as binaries, integers, or strings. This makes it ideal for various positioning methods based on IDs, such as Bluetooth, Bluetooth Low Energy, and Wi-Fi. The approaches can also be used to evaluate the status of vaccination sets, such as those being discussed for the 2020 pandemic. If a general problem can be reduced to a set membership test, the ten approaches can be considered to enable privacy. As the HE approach uses the dot product, i.e., scalar product, it is not suited as described in this thesis, but in combination with the HE-XOR-ZKP scheme, any bitwise comparison of IDs is supported.

Table 7.2: Approaches and objectives

	1: Data send to service	2: Precise target position	3: Fast encoding	4: Service learns what is necessary	5: Geofence not known to service	6: Service may limit geofences	7: Privacy-preserving distance measuring	8: Position shown if outside	9: Arbitrary shaped geofence	10: Feasible memory sizes	11: Geofence updates not necessary (privacy)	12: Low-powered device	13: Geofences may be added, deleted, or updated	14: On-the-fly geofences	15: Separation of positioning and encoding	16: Services do not get shared secrets	17: Support Bluetooth beacons, Wi-Fi signals, etc.
1 Grid cell hashing	●	●	●	◐	●	○	●	○	●	●	◐	●	◐	◐	●	◐	●
2 Probabilistic methods	●	●	●	◐	●	○	○	○	●	●	◐	●	◐	◐	●	●	●
3 Homomorphic encryption	○	●	◐	●	◐	◐	◐	○	●	◐	●	◐	●	◐	●	●	○
4 Multi-party computation	●	●	◐	●	○	◐	◐	○	●	●	●	○	◐	◐	●	◐	●
5 Private set intersection CA,X	●	●	○	●	○	○	○	○	●	○	●	○	◐	◐	●	◐	●
6 Cryptographic accumulators	●	●	●	●	◐	○	○	○	●	●	●	●	●	●	●	●	●
7 Zero-knowledge proofs	●	●	○	●	○	◐	◐	○	●	◐	●	○	◐	◐	●	◐	●
8 Attribute-based encryption	●	●	●	●	◐	○	○	●	●	●	◐	●	●	◐	●	●	●
9 Private pattern matching	●	●	○	●	○	●	◐	○	●	●	●	●	◐	◐	●	●	●
10 Policy-based approaches	●	●	●	○	○	●	◐	●	●	●	●	●	●	●	●	○	●

● Objective is addressed by approach
◐ Objective is partially addresssed by approach
○ Objective is not addressed by approach

7.2.2 Privacy features and LBS classification

Following Table 3.2, in which classes were defined following LBS features and use cases, Table 7.3 gives an overview of the achieved privacy features presented in Chapter 5 and their relation to the classification, Classes 1 to 3. The features can also be compared to the privacy features in current apps summarized in Table 2.1. The presented features can be seen as research additions to privacy functionalities in location-based services.

Table 7.3: Privacy features and LBS classes

Privacy features	LBS classes
Privacy-preserving precise target position evaluation	1, 2, and 3
Service only learns what is necessary	1, 2, and 3
Geofence locations and sizes are not known to the service	1 and 3
Privacy-preserving area limitation	1 and 3
Position of the target only shown if outside privacy area	1
Privacy-preserving geofence updates	2 and 3
No shared secrets are known to the service	3
Privacy-preserving speed measuring	2 and 3
Privacy-preserving distance measuring	3
Privacy-preserving health service information	2
Privacy-preserving LBS based on additional positioning methods	1, 2, and 3

7.2.3 What is known about the target client?

The use cases presented in Chapter 2 primarily address location-based services in which the target is known to the provider, e.g., name, address, date of birth, payment option, and driver's license. This, along with the fact that the position evaluation and the goal of hiding the exact location of the target client while evaluating it against one or more very accurate geofences, challenged the solution approaches presented. Apart from not revealing the client's position, the overall concept of geofence evaluation can disclose additional information to a service.

The following is a list of this additional information that may reveal (aggregated) knowledge about the device or person. This is to be seen as a list of challenges to be overcome with the privacy-preserving techniques discussed in the previous chapters.

The service may know ...

- the start and endpoint of the client,
- when the client moves between position updates,
- how fast the client moves between position updates,
- the mode of transportation,
- when clients are close to each other,
- the time spent in a place (even if it is not known),
- the area where a client has left a geofence,
- the area where a client has entered a geofence,
- whether the client is moving in a large or small grid cell of a geofence,
- how often and for how long a client uses a service,

- when the client is in a traffic congestion, and
- which (type of) street the client is located in.

It can also be seen that some of the information accessed by people using certain services is also dependent on the nature of the service itself. For example, a carsharing service requires that the service knows the transportation mode and the average speed within a city. The solution proposed within the thesis aims to be useful with wireless technologies, i.e., in terms of the client's battery consumption, computing power, and maximum data transfer speeds and volumes.

7.3 Computational evaluation

The following subsections describe the setup for the computational evaluation of different cryptographic approaches.

7.3.1 Geofence representation and encoding

To evaluate the proposed solution, a comparison of S2 and Geohash encoding systems for modeling city-wide geofences and smaller postcode areas covering Berlin and Germany is carried out. The goal is to encode areas with as few cells as possible and optimize the area accuracy in percentage between the actual polygon and its grid cell representation.

7.3.1.1 Seven German cities

The evaluated cities are the seven German cities with active carsharing companies; Berlin, Düsseldorf, Frankfurt (am Main), Hamburg, Cologne, Munich, and Stuttgart. Table 7.4 and Figures 7.1 and 7.2 show how many S2 cells and Geohashes are needed to cover almost 100% of the area of each individual city. On the one hand, the number of Geohashes needed to cover each city is less than the number of S2 cells. On the other hand, the accuracy of the area covered with S2 cells is much closer to 100% with S2 cells than with Geohashes. This is because an S2 cell has four children (compared to 32 for Geohash) and can cover the entire area more densely.

Table 7.4: Seven cities with a number of S2 cells to cover an area with 100% accuracy.

City	km^2	#S2 cells	#Geohashes
Berlin	891.12	7,912	5,769
Düsseldorf	217.41	8,806	2,753
Frankfurt	248.31	8,616	2,803
Hamburg	755.09	7,227	5,241
Cologne	405.01	4,955	3,390
Munich	310.71	8,664	2,635
Stuttgart	207.35	7,528	1,975
Sum	3,035.00	53,708	24,566

Figure 7.3 shows the same result for Berlin's postcode area 10587 around the Ernst-Reuter-Platz. S2 covers the area more accurately but at the expense of grid cells. The area can be covered by 2371 S2 cells at 101.33% accuracy or 1190 Geohashes at 108.00%. The number of S2 cells to cover an area towards 100% accuracy increases rapidly as smaller cell levels

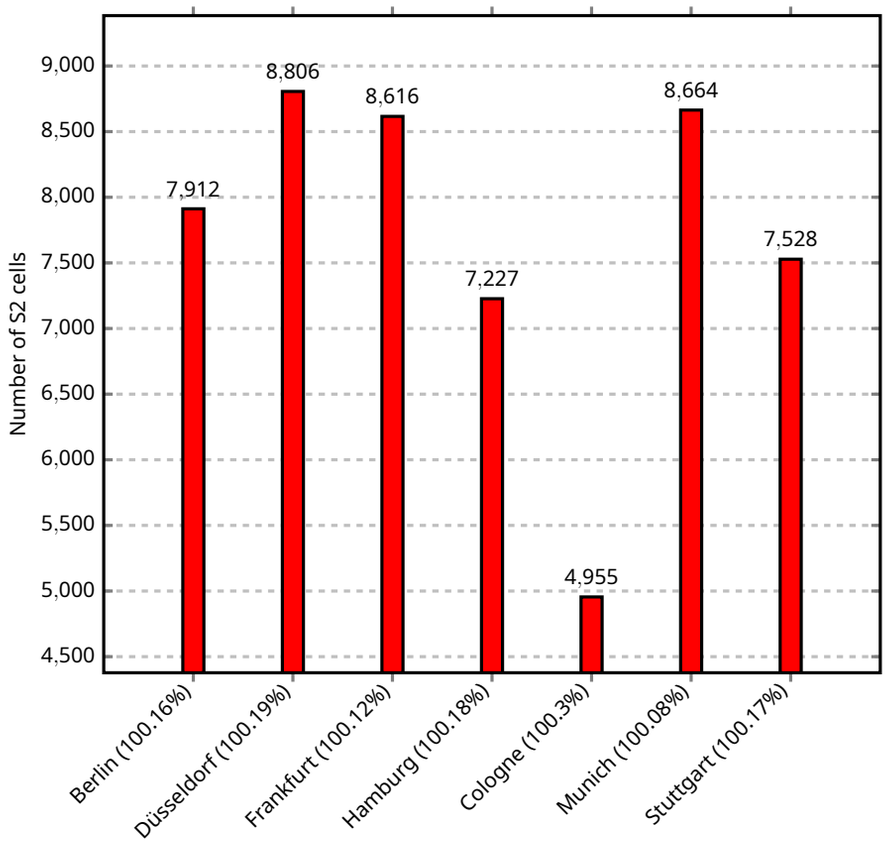


Figure 7.1: Seven cities with a number of S2 cells to cover an area with 100% accuracy.

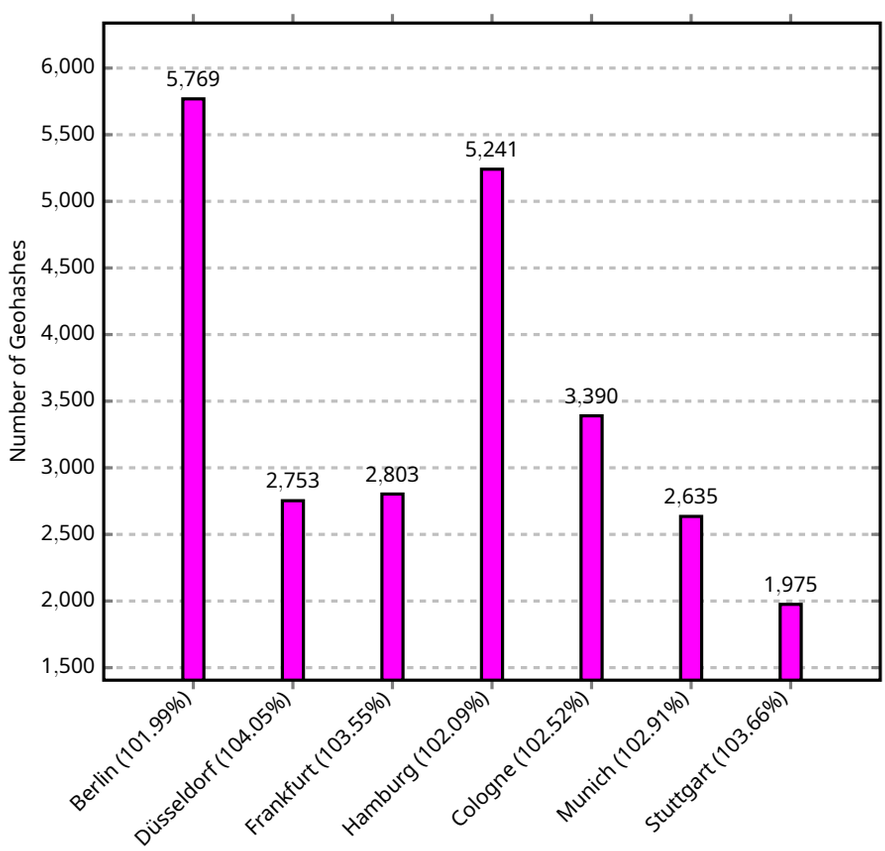


Figure 7.2: Seven cities with amount of Geohashes to cover the area around 100% accuracy.

are required. The numbers are above 100% because the genuine city areas are always completely covered by the cells, and adding up all cells increases the covered area. Both geo encodings are suitable for the presented cryptographic approaches presented. The use cases themselves demand the required accuracy.

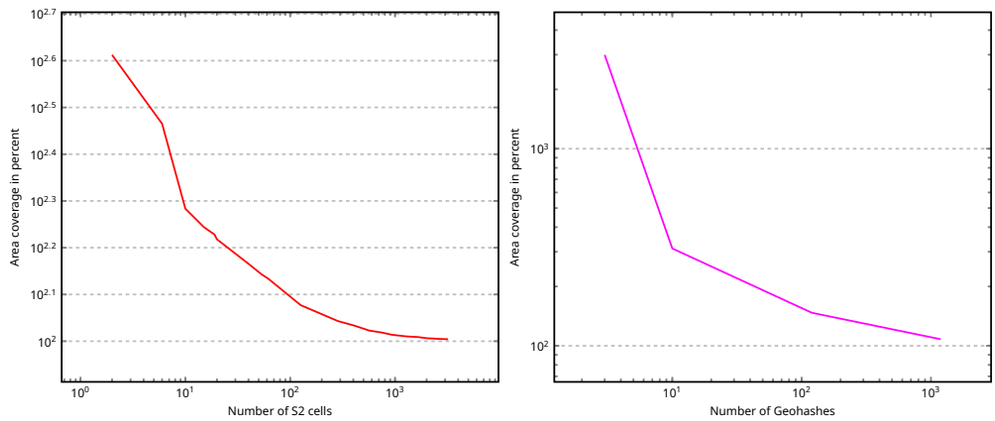


Figure 7.3: The number of S2 cells (left), Geohashes (right) and area coverage in percent compared to the original postal code area 10587 at Ernst-Reuter-Platz in Berlin, Germany. Both axes use a logarithmic scale.

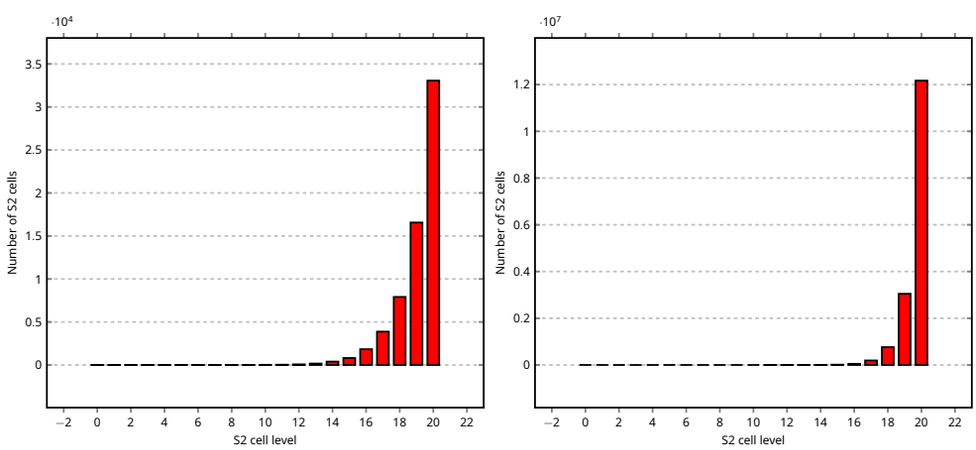


Figure 7.4: Number of S2 cells covering Berlin with different maximum values (normalized and non-normalized).

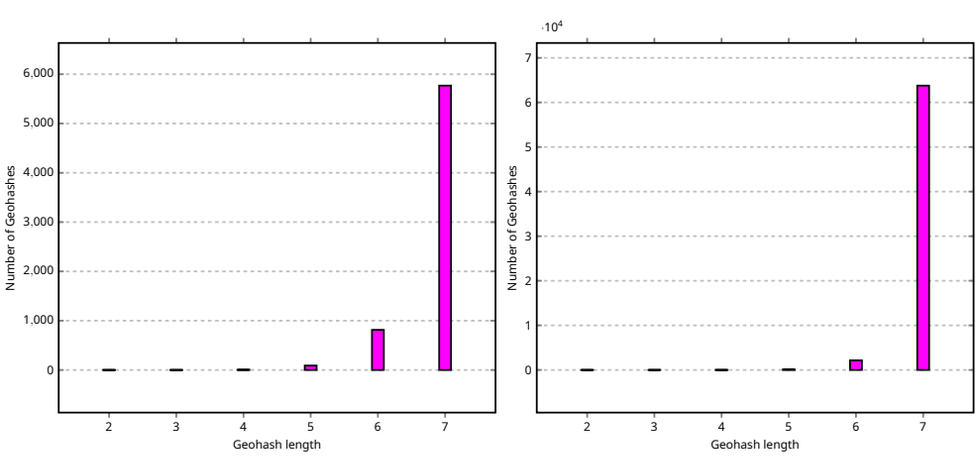


Figure 7.5: Number of Geohashes needed to cover Berlin with different max. levels (normalized and non-normalized).

Figures 7.4, 7.5, and 7.6 show how the number of grid cells depends on the use of the normalized form (parent cells are used instead of all child cells) and the non-normalized form (only one grid cell level is used). The figures show the cell levels for Berlin and

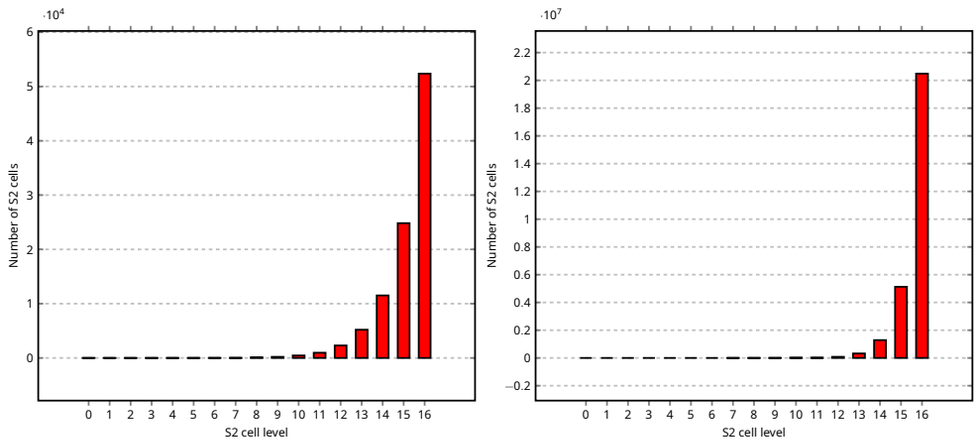


Figure 7.6: Number of S2 cells needed to cover Germany with different max. levels (normalized and non-normalized).

Germany with S2 and Berlin with Geohashes. Since many small grid cells are needed to cover an area, the non-normalized numbers are significantly higher, especially for S2 cell levels of 16 (Germany) and 20 (Berlin). Examples of the normalized and non-normalized representation for the Berlin geofence were shown in Section 5.2.

7.3.1.2 Inverse Berlin example

The notion of inverse geofences was introduced in Section 5.2.2. Figures 7.7 and 7.8 show the coverage of Berlin by normalized S2 representation (810 cells) and by their normalized inverse representation (896 cells).

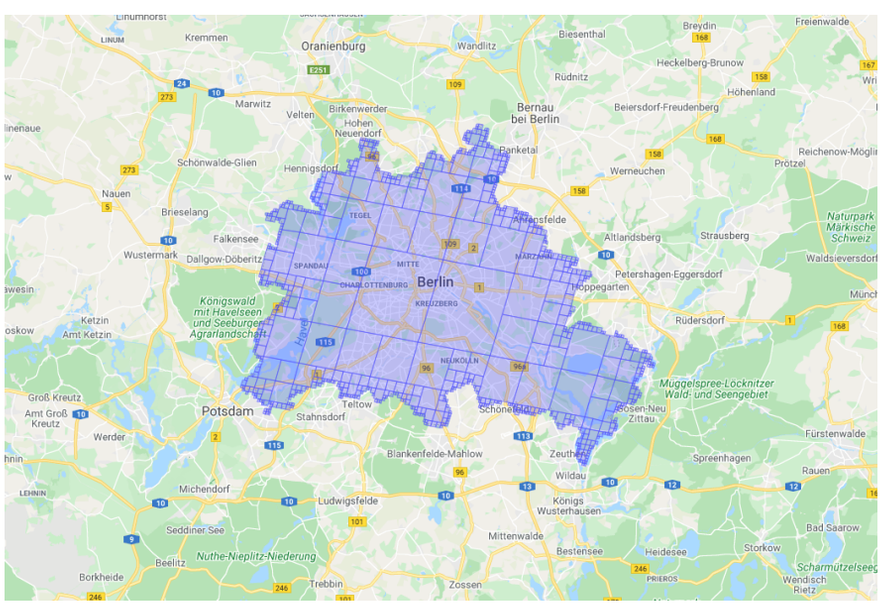


Figure 7.7: Berlin represented with 810 S2 cells (covering geofence).[1]

Table 7.5 shows the number of S2 cells needed for the regular and inverse representation of the Berlin geofence at different cell levels. Using this table and Figure 7.9, there is little difference in the number of cells needed to represent a geofence with both inner and

[1] Image source in this subsection: Figures created using the QGIS open source application, underlying map data by Map data ©2018 GeoBasis-DE/BKG (©2009), Google.

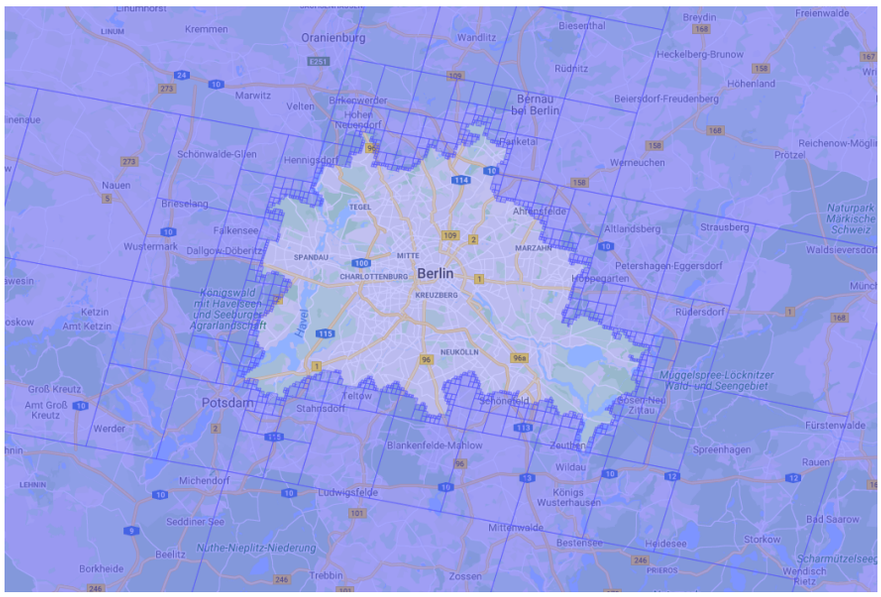


Figure 7.8: Berlin represented with 896 S2 cells (inverse, not covering geofence).

outer grid cells. For example, at S2 level 21, 67,487 cells are necessary for the normal geofence and 67,589 for its inverted representation (normalized and based on the same Berlin polygon). This suggests that the cryptographic approaches and use cases based on inverse geofences, such as the ABE approach, do not need to compute a more significant number of cell values.

Table 7.5: Number of S2 cells of Berlin inside and outside (inverse)

S2 cell level	#S2 cells	#Inverse S2 cells
10	19	37
11	32	51
12	71	124
13	165	201
14	392	424
15	810	896
16	1,827	1,869
17	3,906	3,946
18	7,930	8,076
19	16,647	16,374
20	33,219	33,702
21	67,487	67,589
22	135,656	136,522
23	273,105	272,687
24	547,748	545,956
25	623,722	623,053

7.3.1.3 Berlin postcode areas

Berlin consists of over 190 arbitrary-shaped postcode areas, represented by a five-digit number (1****) starting with 10, 12, 13, or 14. To evaluate the privacy approaches, 191 polygons were converted into normalized grid cells with different levels of accuracy.

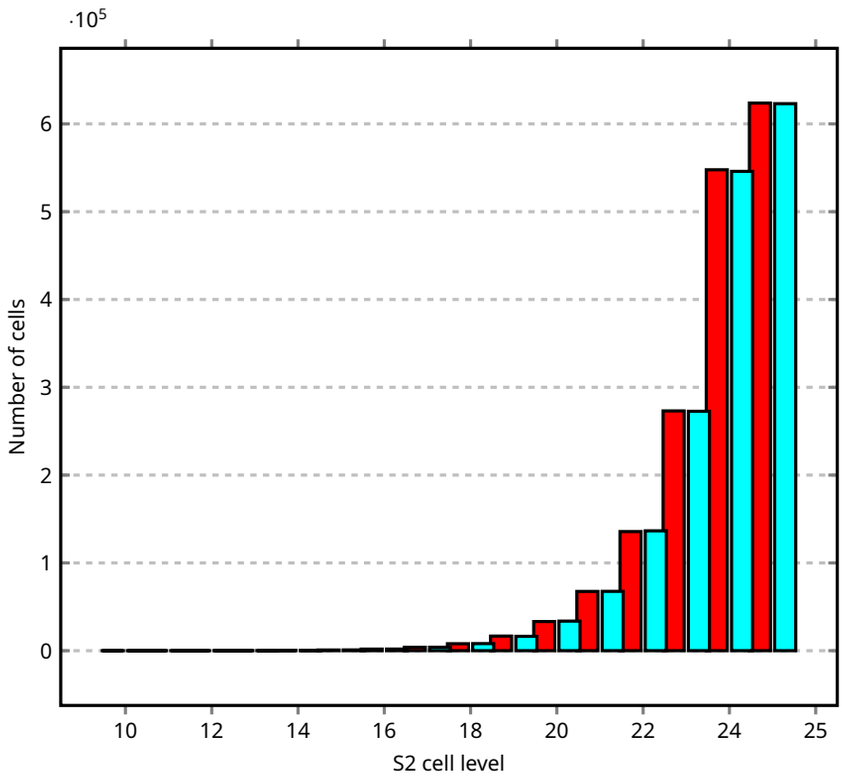


Figure 7.9: Comparison between the number of S2 cells for a geofence representation of Berlin (red) and its inverse geofences (cyan) at different S2 cell levels.

Due to the limitations of the tools used for privacy-preserving geofencing, a compromise had to be found between the accuracy level and the amount of data processed. For the length 9 (nine characters of a Geohash), the error is around ±2.4meters. This value can be considered acceptable for the presented use cases. For the Geohash encoded postcode areas, lengths 4 to 9 were used in the evaluation. It will be seen that even length 9 generates too much data for its representation with some cryptographic tools, so lower precision levels were used in these cases.

With the S2 encoding, there are four children for each parent grid cell. For the analysis, S2 cell level 21 was chosen, which fits the use cases with an average area of 19.33m^2. As with the Geohash data, not all tools could process it, so less precise levels were used. In the test cases, the normalized form of the geo encoded polygons was used within the test cases. Figure 7.10 shows the postcode areas of Berlin [250]. Figure 7.11 shows the number of grid cells needed to cover postcode areas of Berlin.

7.3.1.4 Geofence generation duration

Figure 7.13 shows the time required to compute the geofences with different precision levels. As explained earlier, the S2 cell and Geohashes encodings take an exponential amount of time to create, as the precision levels increase by either 4 or 32 for each additional level. The figure shows that the maximum precision level is 21 for S2 and 9 for Geohashes. The tested cryptographic tools that use the postcode areas ranges include the grid cell hashing approach presented in this thesis, Section 7.3.2, the homomorphic

[2]Image source: Data used to produce the figure is from https://www.suche-postleitzahl.org/.

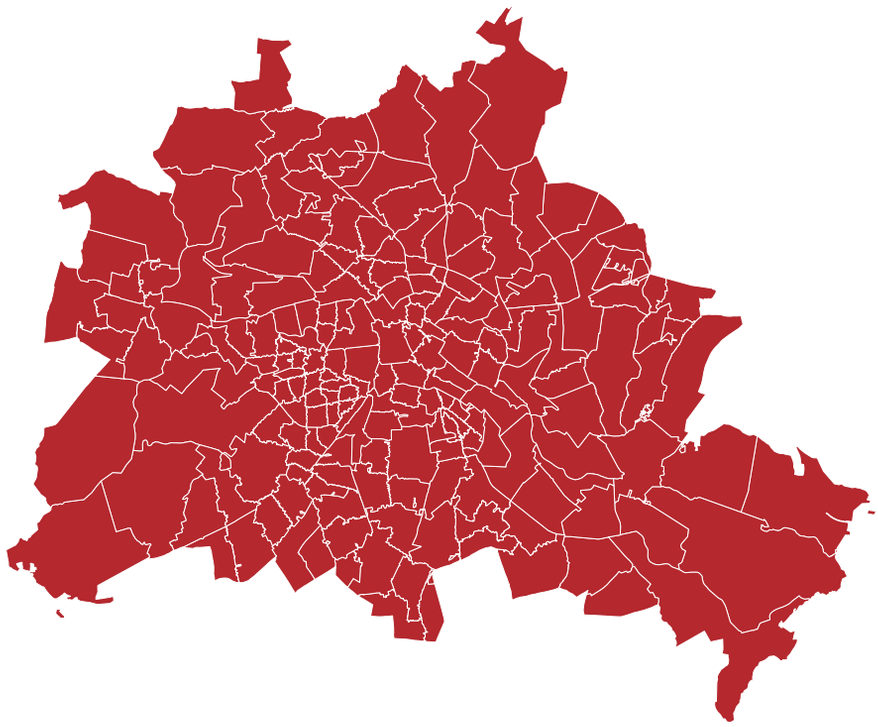


Figure 7.10: The five-digit postcode areas of Berlin (Germany).[2]

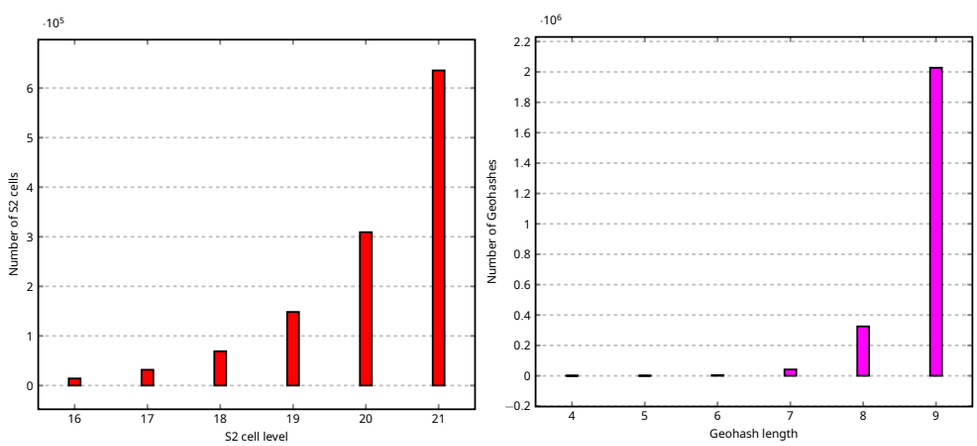


Figure 7.11: Number of S2 cells and Geohashes to represent Berlin's postcode areas using different precision levels.

encryption approach NEXUS and its parallel equivalent, Section 7.3.4, and the Pequin ZKP, Section 7.3.5.

Test environment The following specification evaluated the above cryptographic approaches using postcode areas on a MacBook Pro.

- 3.1 GHz Intel Core i7 CPU, including eight logical cores
- 16 Gigabyte LPDDR3 RAM at 2133 MHz

Due to the power of containerization, the evaluation framework was tested with different operating systems. The container environment also works with Windows 10 and Ubuntu 18.04. The tests were executed independently. Since the tools were executed inside Docker images, the specifications of the containers are also relevant. The Docker environment has the following specifications.

- CPU with eight cores
- 10 Gigabyte RAM
- No swap area

The swap area was set to zero to limit the memory consumption to 10 Gigabytes.

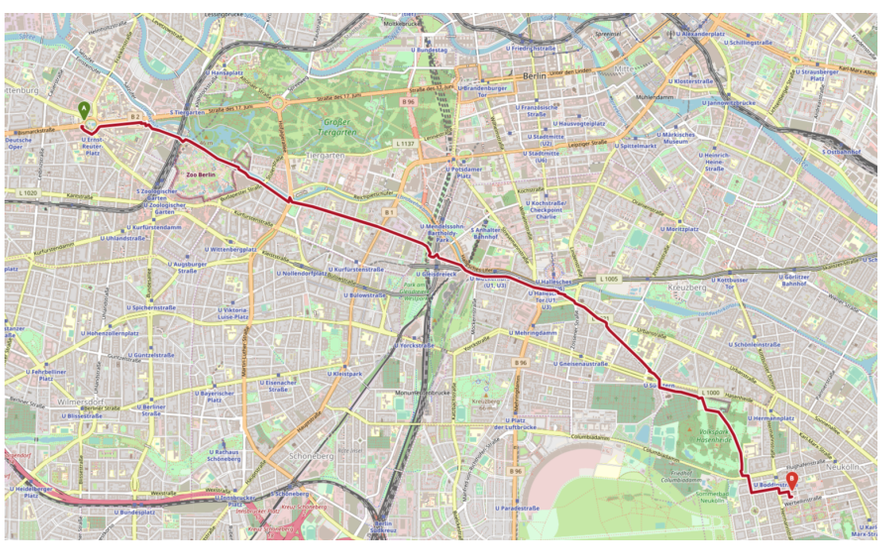


Figure 7.12: Driving route from Ernst-Reuter-Platz to Rollberg cinema in Berlin.[3]

Test coordinates Two types of location points are used for the tests and the described setup. One consists of 400 coordinates representing a driving route from Ernst-Reuter-Platz to the Rollberg cinema in Berlin, Figure 7.12. Another consists of 400 random points in Berlin. These location points are also represented by coordinates, S2 cells, and Geohash values. Due to the computational limitations of the test hardware, not all levels are used. Precision levels are used from the lowest reasonable to the highest processable levels. Berlin postal code areas are used as geofences, with each area being shown separately.

The difference in the exponential growth of the two geo encodings explains the increased Geohash representation over time between lengths 8 and 9, Figure 7.13. The number

[3]Image source: The underlying map data is provided by OpenStreetMap (OSM) under the Open Database License (ODbL), openstreetmap.org/copyright.

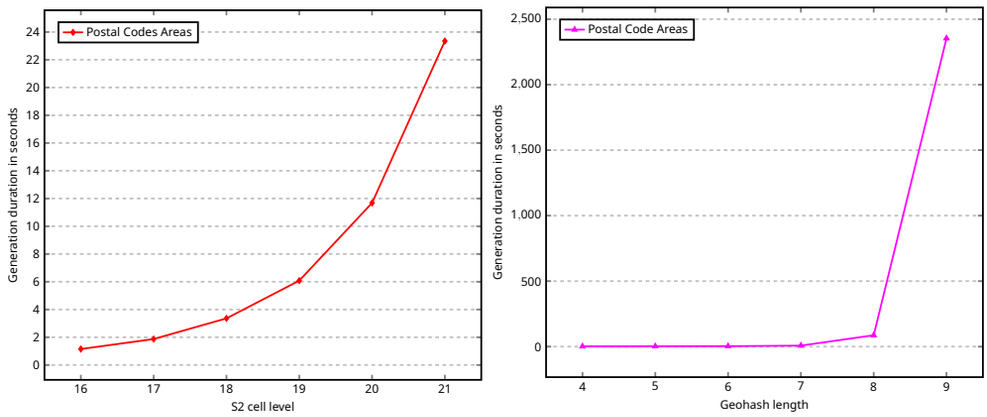


Figure 7.13: Geofence generation durations for all postcode areas as S2 cells (red) and Geohashes (magenta)

of Geohashes representing each postcode increases by a factor of 32 instead of 4 in the case of the S2 representation. It can be seen that the encoding of S2 values is done much faster than the generation of Geohash-based geofences. It should be emphasized that the values shown apply to the generating of all 190 postcode areas within the Berlin polygon as described above.

7.3.1.5 Geofence loading memory consumption

After generating the grid cell encoded geofences, they are loaded into the memory on the server-side. The following three diagrams show this memory consumption in megabytes for each grid cell level for both geo encodings. The values for the Pequin tool are used on the client-side because the geofence encoding must be present there. The missing values for some tools in the diagram are the poor performance of some tools at high numbers of data, i.e., there are too many grid cells to represent the more accurate geofence.

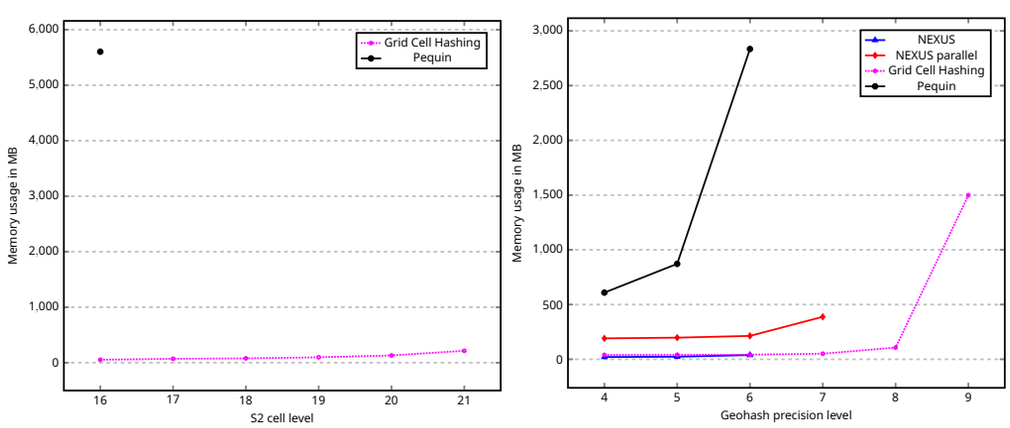


Figure 7.14: The maximum memory consumption when loading geofences is shown in separate graphs for S2 cells and Geohashes. Except for Pequin, this part is done server-side. The color/shape codes used in this and the following graph are: NEXUS (blue triangle), NEXUS parallel (red diamond), grid cell hashing (magenta asterisk), and Pequin (black dot).

The diagram in Figure 7.15 shows the combined memory consumption when loading the grid cells onto the server, for Penquin onto the client. The x-axis indicates the number of cells required. The higher the number of cells, the more accurate the geofence representation is.

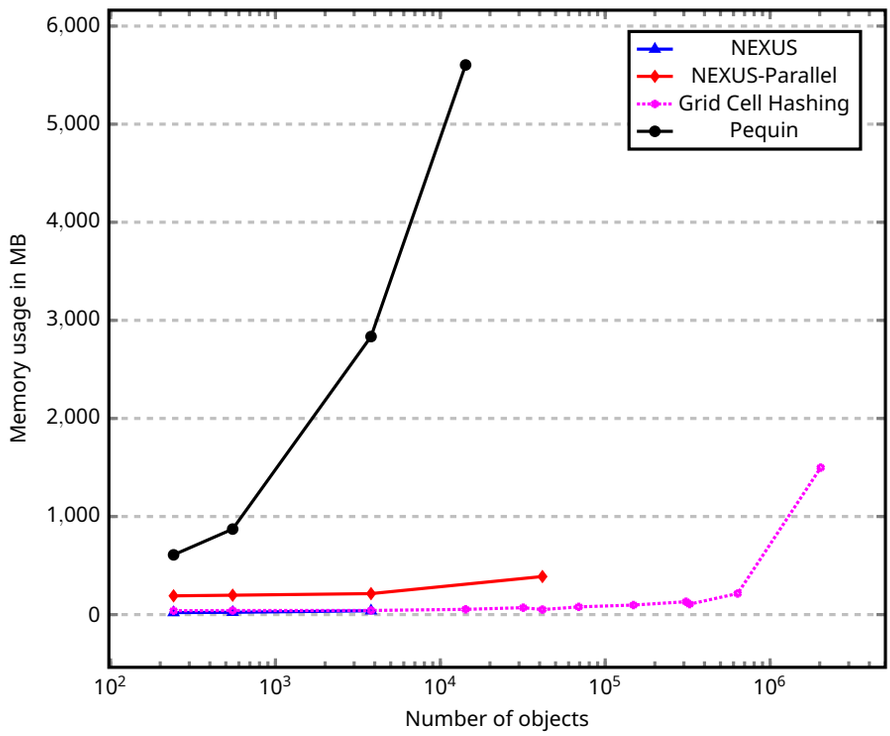


Figure 7.15: The maximum memory consumption when loading geofences is shown in a combined graph to show the relationship between memory consumption and the different number of S2 or Geohash objects

7.3.1.6 Geofence loading duration

The different approaches calculated how much time it takes to load the geofences into the system, i.e., the database or the tool. The first two graphs in Figure 7.16 show the precision levels on the x-axes and the duration for loading the geofences on the y-axes. As with memory consumption, it is essential to mention that the loading takes place on the client-side in the case of the Penquin tool. In addition, its capabilities were limited because the program crashed for higher precision levels. Therefore, the values for the Geohashes are measured only up to length six and, in the case of S2 cells, up to level 16.

Based on the diagrams, it is clear that the hash values perform best, even at high precision levels, because they only store values within a database. The original NEXUS approach performs less well than its paralleled version. Nevertheless, even with the paralleled version of NEXUS, it still takes about 50 minutes to load all the geofences, and that is only with a length of 7 for the Geohashes.

The combined chart in Figure 7.17 shows the number of objects on the x-axis processed by the different approaches.

Creating the geofences and their (cryptographically) geo encoding occurs during the setup phase of various use cases. In this respect, an extensive loading time for the geofences is acceptable. In the evaluation case, 190 postcode areas and Berlin itself were encoded. Germany's cities, municipalities, and regions are divided into 13108 different five-digit postal code areas. This number is about 70 times higher than the number of postcode areas evaluated. To compile all of these with the slowest tool, the NEXUS-Parallel with a Geohash precision length of 7, used for the privacy-aware geofencing, would mean that the compilation would take about 58 hours on the system described above.

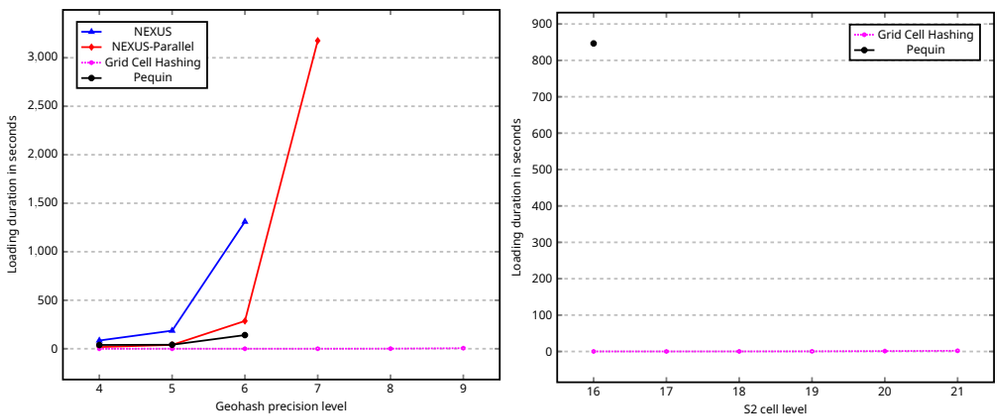


Figure 7.16: Geofence loading durations for different S2 and Geohash precision levels

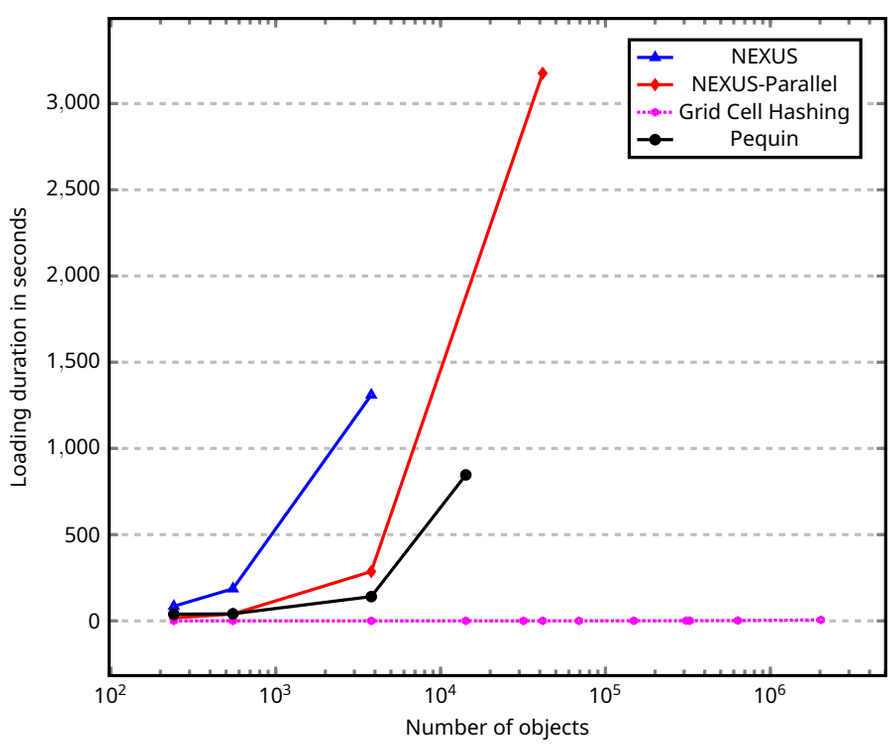


Figure 7.17: Geofence loading durations for different number of S2 cells or Geohash objects

7.3.2 Grid cell hashing

The section describes the performance evaluation of the first approach introduced in Chapter 5.

7.3.2.1 Cryptographic hashing algorithms

The first test includes various cryptographic algorithms and grid cell values representing Germany. The results show that the computational time only progresses linearly as a function of the number of grid cells. When comparing the hashing durations, the influence of salt and nonce values, such as a timestamp, can be neglected.

Testing environment Some data was compiled based on the testing environment mentioned in Section 7.3.1.4. Additional tests were performed based on the following values.

- MacBook Pro (13-inch, 2017, Two Thunderbolt 3 ports)
- 2,3 GHz Intel Core i5
- 16 Gigabyte 2133 MHz LPDDR3 RAM

The high-accuracy polygon for Germany [251] for the following calculations is a GeoJSON file [252] with 2.445.312 bytes (2.4MB).

The following software and packages were used.

- Python 3
- hashlib
- argon2
- pbkdf2

Figure 7.18 and Tables 7.6 and 7.7 evaluate the computation times for different hashing algorithms and different sets of S2 cells (relative to the Germany polygon). The seven hashing algorithms and configurations are as follows.

- 1: Only one-time hashing using SHA512
- 2: Hashing(string+currenttime(HMS)) using SHA512
- 3: Password-based key derivation function using PBKDF2
- 4: Hashing only once using Argon2(pw_string+salt)
- 5: Hashing(string+currenttime(HMS)) using Argon2
- 6: Hashing only once (Argon2, password=i, salt=some salt, t=16, m=8, p=1, buflen=128, argon_type=argon2.Argon2Type.Argon2_i)
- 7: Hashing(string+currenttime(HMS)) using Argon2, password=i+time, salt=some salt, t=16, m=8, p=1, buflen=128, argon_type=argon2.Argon2Type.Argon2_i)

Figure 7.18 shows that the computation times are linear. It also shows that SHA512 hashing takes only 69ms or 255ms, respectively, even with over 50,000 cells. Modern hardware is optimized for these operations because hashing is such an elementary calculation. More complex hashing algorithms, such as Argon2, which include timestamps and salt values, compute large numbers of S2 cells in hundreds of seconds. For use cases where the target sends position updates, SHA512 hashing can be used every few seconds for privacy preservation. Use cases with less rapid position updates can use other KDFs and cryptographic hashing algorithms.

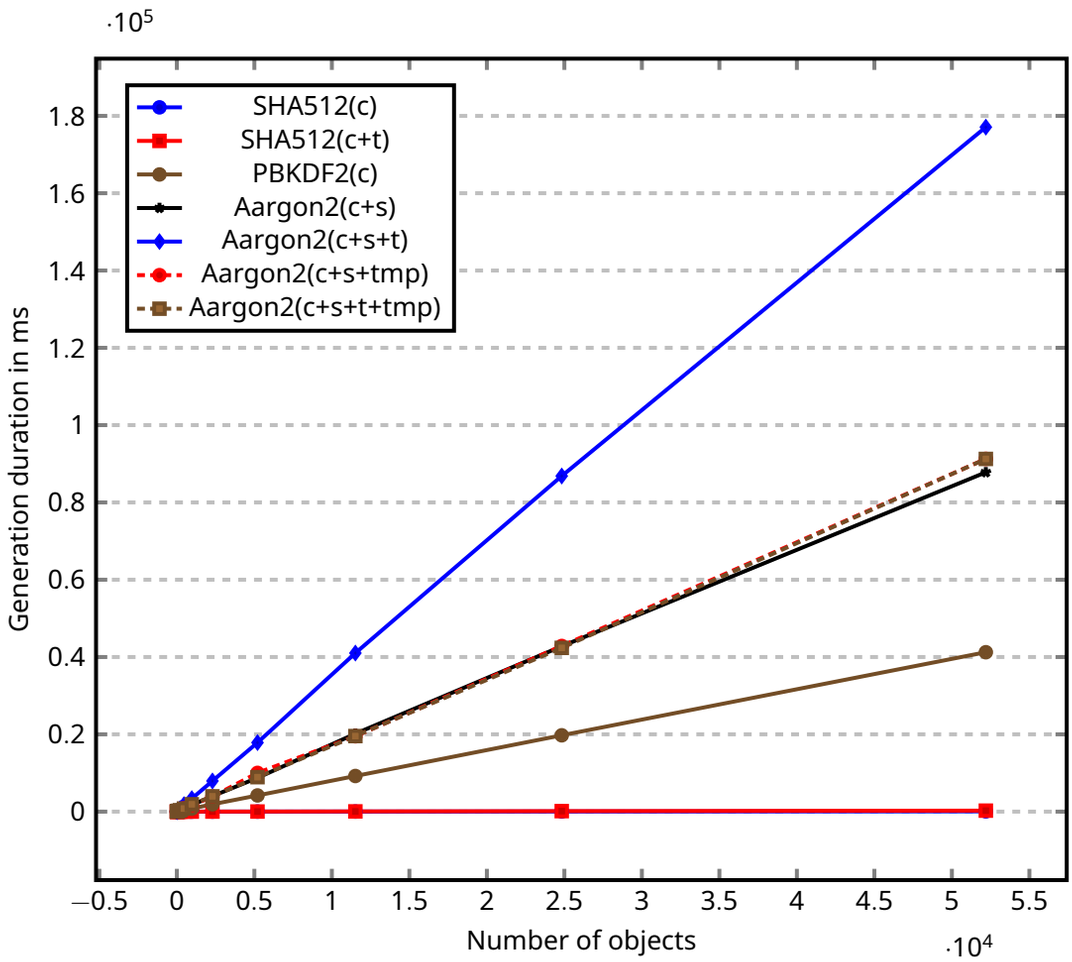


Figure 7.18: Object hashing durations in ms for different algorithms of S2 cells representing Germany

Table 7.6: S2 hashing Germany (times are in milliseconds)

#Cells	SHA512(cell)	SHA512(cell+time)	PBKDF2(cells)
5	0.15	0.25	4.26
7	$4.49 \cdot 10^{-2}$	$7.2 \cdot 10^{-2}$	5.6
12	$4.93 \cdot 10^{-2}$	0.14	11.28
25	0.18	0.7	21.35
95	0.23	0.89	81.49
204	0.32	1.03	168.36
456	0.64	2.25	367.66
964	1.33	4.69	784.96
2,296	3.06	12.99	1,903.87
5,203	6.92	25.36	4,186.98
11,517	15.45	59.37	9,216.53
24,830	33.68	129.71	19,745.05
52,191	69.16	255.24	41,242.5

Table 7.7: S2 hashing Germany (times are in milliseconds)

#Cells	Aargon2(c+s)	Aargon2(c+s+t)	Aargon2(c+s+tmpbi)	Aargon2(c+s+t+tmpi)
5	12.11	26.66	10.27	8.57
7	22.03	37.09	16.24	17.03
12	30.86	62.95	22.03	34.14
25	58.59	115.11	56.08	60.89
95	186.56	367.95	211.83	227.09
204	416.14	787.57	378.82	359.79
456	794.12	1,804.33	865.37	800.96
964	1,687.82	3,359.27	1,662.76	1,896.96
2,296	4,000.7	7,933.17	3,891.98	3,956.46
5,203	8,737.28	17,856.9	10,022.7	8,917.14
11,517	20,160.88	41,024.63	19,493.63	19,547.17
24,830	42,780.6	86,833.51	42,855.53	42,366.64
52,191	87,791.18	$1.77 \cdot 10^5$	91,264	91,254.65

7.3.2.2 Carsharing and SHA512 database sizes

Tabel 7.8 shows the number of carsharing vehicles and the number of registered users in Germany, in 2021 [253]. It forms the basis for calculations of the required database space for storing hash values.

Table 7.8: Amount of carsharing vehicles and registered users

Type	Amount
Station-independent cars	14,200
Station-based cars	12,020
Station-independent registered users	2,150,300
Station-based registered users	724,100
Vehicles sum	26,220
Users sum	2,874,400

The following multiplications are performed to calculate how much data is needed to store hash values.

- Number of S2 cells in the seven largest carsharing cities 53,708 (as of Section 7.3.1, Table 7.4)
- Multiply by the number of station-independent vehicles = 762,653,600
- Multiply by 512 bits (SHA512) = $3.90 \cdot 10^{11}$ bits = 48.75 gigabytes

Table 7.9 summarizes the database sizes for the grid cell hashing approach based on SHA512. Even for use cases that include all registered cars in Germany and S2 cells covering the German national road network, feasible database sizes, i.e., 79.24 TB, can be achieved for hash values of 512 bytes. As increasing storage sizes become more affordable, the calculations show reasonable values for the presented use cases.

Table 7.9: Database sizes for GCH (SHA512) with numbers for Germany

#S2Cells	#Station-indept vehicles/#carsharing users	Memory size
7,912 (Berlin)	14,200 cars	7.19 GB
53,708 (7 cities)	14,200 cars	48,75 GB
53,708 (7 cities)	2,150,300 users	7.39 TB
GER street network	47,096,000 vehicles in GER	79.24 TB

7.3.2.3 Comparable calculations

In this subsection, the GCH approach evaluates the memory consumption for proof generations, the duration of proof generation, and the duration of verification. These tests are either based on different cell levels for the S2 and Geohash encodings or the overall objects.

Setup and tests The following figures are based on the test environment described in Section 7.3.1.4. The SHA512 hashing algorithm is used for all tests. Figures 7.19 to 7.23 show the results. This is reflected in the *proof generation* and *verification* processes. In the LBS execution phase, the client generates a *proof* sent to the service. At the service, this *proof* is *verified*. In the proof, the current position is hashed with additional salt and time information, and this data is sent to the service. During verification, the service performs hash operations on the geofence data from the *setup* and compares the received values with the geofence(s) values.

- Memory consumption of the proof generation in MB is based on the precision levels (S2 and Geohash).
- Memory consumption of the proof generation in MB is based on the number of objects (S2 and Geohash).
- Duration of the proof generation in seconds based on the precision levels (S2 and Geohash).
- Duration of the proof generation in seconds based on the number of objects (S2 and Geohash).
- Verification time in seconds based on precision levels (S2 and Geohash).
- Verification time in seconds based on the number of objects (S2 and Geohash).

To compare memory consumption and durations, the same calculations are performed for the homomorphic encryption approach NEXUS, including its paralleled modification and the zero-knowledge proof Pequin library, see Sections 7.3.4 and 7.3.5).

Figures 7.19 and 7.20 show the memory consumption needed for the GCH scheme. Depending on the number of objects to be hashed (also on the server-side), the proof generation process consumes 50 to 450 MB. The client only needs to generate 30 values which explains its short runtime in Table 7.21.

The GCH verification involves comparing geofence hash values on the server side with the generated proof values. Figures 7.22 and 7.23 show that this process is very short, under 0.5 seconds, even for a high number of objects (up to 2,027,738 values).

The memory consumption and the duration of the GCH approach differ significantly from the HE and ZKP schemes instantiated directly with the same evaluation environment.

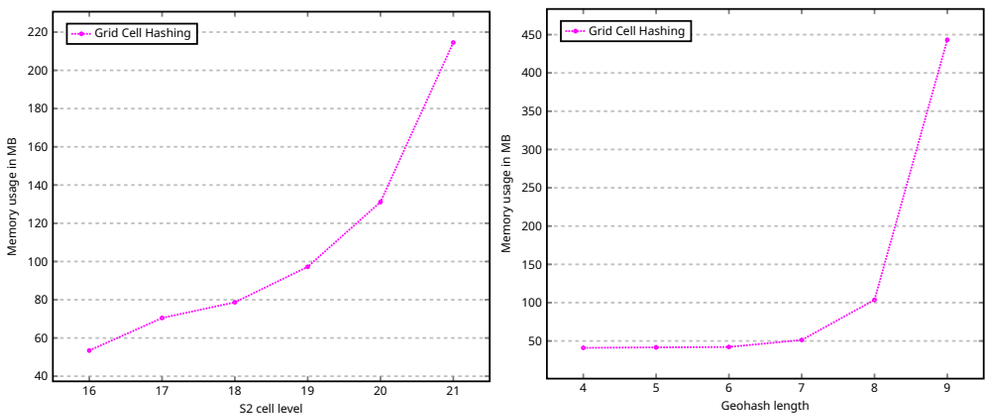


Figure 7.19: Maximum memory consumption in MB for evidence generation for S2 cell level 16 (left) and different Geohash lengths (right). The magenta line represents the SHA512 cryptographic hash algorithm.

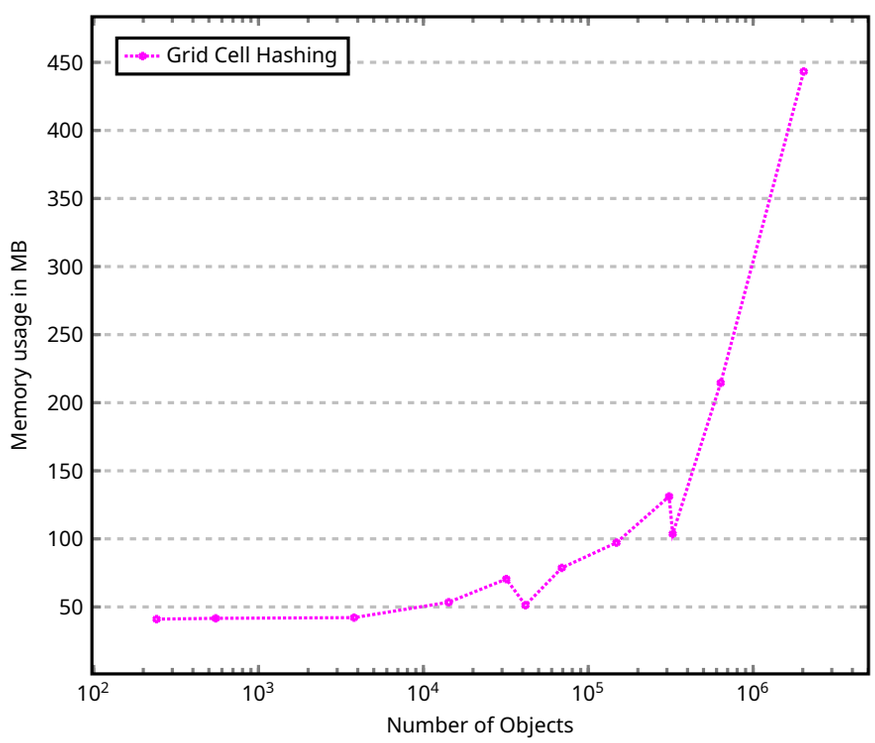


Figure 7.20: Maximum memory consumption in MB for proof generation depicted for different object counts (both S2 cells and Geohashes). The magenta line represents the SHA512 cryptographic hashing algorithm.

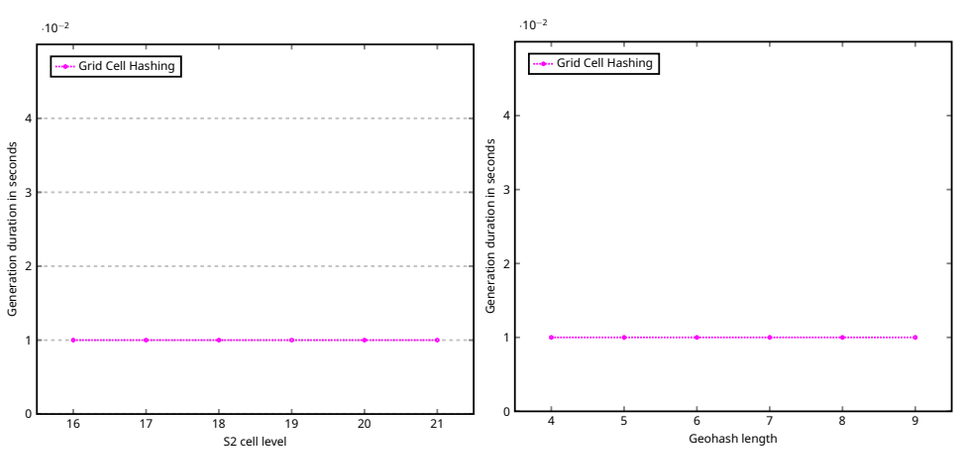


Figure 7.21: Proof generation durations in seconds depicted for S2 cell levels (left) and Geohash lengths (right). Proof generation only uses parent cell values and is, therefore, very fast at 0.01 seconds. The magenta line represents the SHA512 cryptographic hashing algorithm.

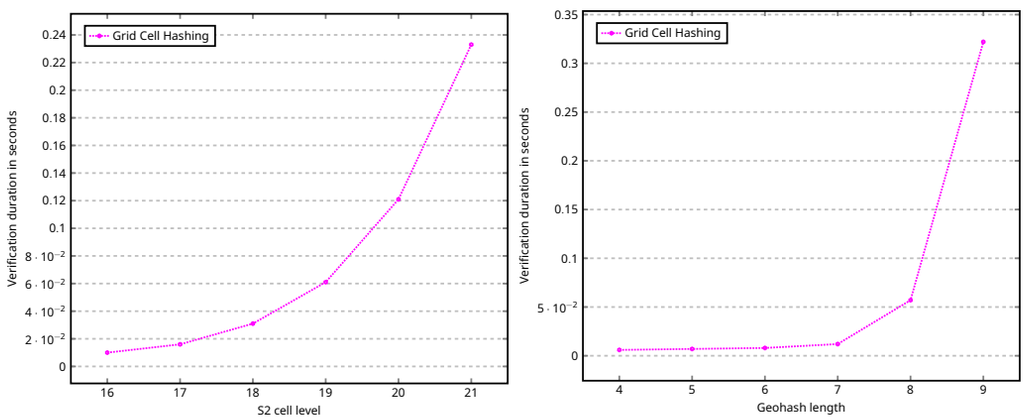


Figure 7.22: Verification durations in seconds for different S2 cell levels (left) and Geohash lengths (right). Verification compares parent cell values with geofence values and is, therefore, also very fast at max. 0.322 seconds. The magenta line represents the SHA512 cryptographic hashing algorithm.

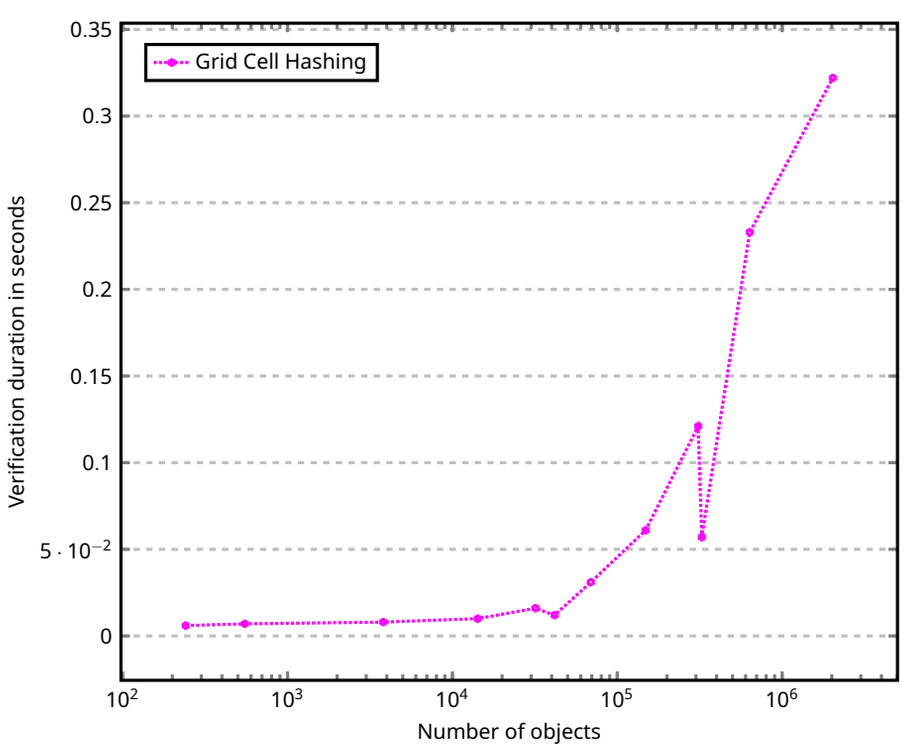


Figure 7.23: Verification durations in seconds for different object counts. Verification compares parent cell values with geofence values and is, therefore, also very fast at max. 0.322 seconds. The magenta line represents the SHA512 cryptographic hashing algorithm.

7.3.3 Probabilistic methods

The numbers presented in this subsection reflect the memory sizes of the Bloom filter, depending on the number of grid cell elements for S2 cell levels 16-20, the number of hash functions *k*, and the *accepted* false positive (FP) rate.

Table 7.10: Number of items depending on S2 cell level

Area name	km^2	Level 16 (153m)	Level 18 (38m)	Level 20 (10m)
Berlin	891.12	1,835	7,912	33,061
7 Cities	3,035		53,708 (max 18/19)	
Germany	357,386	6,324,378 (expol.)		
Street network	16,333	289,032 (expol.)		

The areas used for these calculations are based on the figures presented in Section 7.3.1. The figures for Germany and the German road network are extrapolated in Table 7.10.

Table 7.11: Bloom filter size depending on k and FP rates

Area name	#Items	k=10, FP=0.001	k=13, FP=0.0001
Berlin	7,912	13.9 kB	18.5 kB
7 cities	53,708	94.3 kB	125.7 kB
Cars	483,372,000	848.36 MB	1131.31 MB
Streets+vehicles	$700 \cdot 10^{12}$	1,253 TB	1,638 TB

These values are used to calculate the storage sizes in Table 7.11. The largest memory required is 1.638TB for instantiating a Bloom filter, with k=13 and an FP=0.001, based on the area of the road network and the number of registered cars. This can be considered a manageable storage size for a service.

Table 7.12: Items n depending on accuracy

Area name	km^2	1,111m	111m	11.1m
Ulm	118.69	9,616	963,315	96,331,467
State of BW	35,751	30,585	3,063,956	306,395,585
Germany	357,386	289,541	29,006,226	2,900,622,596

Table 7.13 shows the Bloom filter memory sizes for three different cell sets.

Table 7.13: Bloom filter size depending on k and FP rates

#Cells	k=10, FP=0.001	k=13, FP=0.0001
1,000	1.75 kB	2.34 kB
1,000,000	1.71 MB	2.29 MB
100,000,000	171.39 MB	228.53 MB

7.3.4 Homomorphic encryption

The following figures are based on the test environment described in Section 7.3.1.4. The results here are directly comparable with Section 7.3.2.3.

In this section, two HE schemes are presented. The NEXUS approach has been modified from its original version [176], as described in Section 6.1.3. The second HE approach, NEXUS-Parallel, is the one described in the concept chapter. Both utilize Geohashes and grid cell encoding to enable arbitrarily shaped geofences.

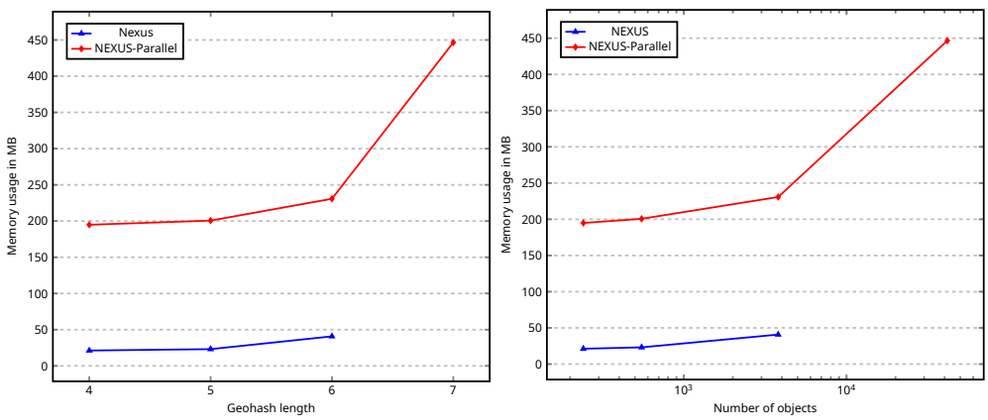


Figure 7.24: Maximum memory consumption in MB for proof generation depicted for different Geohash lengths (left) and number objects (right). The blue line represents the adapted NEXUS scheme, and the red line represents its paralleled version. The object count for S2 cells was too high to get reasonable results in the testing environment.

Since these HE schemes use a trusted third party for an encrypted geofence evaluation, the computation is considered to reside on a server. Figure 7.24 shows the memory consumption required to generate HE proofs. The parallel version consumes significantly more megabytes than its standard version. Due to performance issues, the standard version only allowed Geohashes of length six and below. The calculations for the paralleled version include level 7. Its memory consumption is comparable to that of the GCH approach, requiring memory sizes of around 450MB.

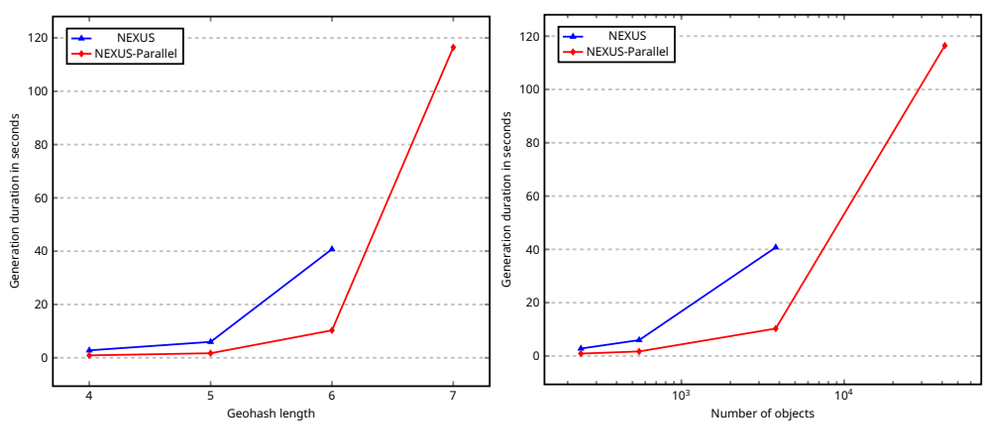


Figure 7.25: Proof generation durations in seconds for different Geohash lengths (left) and the number of objects (right). The blue line represents the adapted NEXUS scheme, and the red line represents its paralleled version. The latter version performs better and is nearly twice as fast in the proof generation. The object count for S2 cells was too high to get reasonable results in the testing environment.

The proof generation times shown in Figure 7.25 would significantly impact the usage of an HE scheme if the use case requires rapid position updates. With 41,611 objects, the proof generation time takes about 120 seconds.

The verification times shown in Figure 7.26 are almost twice as long as the times for proof generation times. This may be aimed at similar use cases.

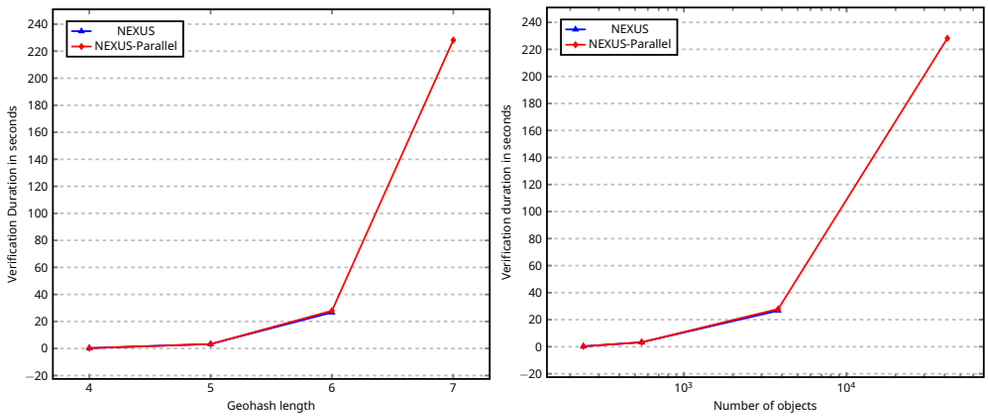


Figure 7.26: Verification durations in seconds for different Geohash lengths (left) and the number of objects (right). The blue line represents the adapted NEXUS scheme, and the red line represents its paralleled version. Verification times of both versions are similar, up to an object count of 3,800. The object count for S2 cells was too high to get reasonable results in the testing environment.

7.3.5 Zero-knowledge proofs

In this section, different ZKP libraries are evaluated. First, the Pequin library is compared using the same evaluation setup as in the sections on GCH and HE. Second, the ING library is presented based on set membership tests and range proof.

7.3.5.1 Pequin scheme evaluation

The ZKP toolchain Pequin [237] was presented in the implementation work and implementation chapters. It utilizes the ZKP scheme zkSNARK [254].

Figures 7.27 and 7.28 show the memory consumptions for different cell levels or object numbers. The Pequin ZKP library consumes the most memory than its GCH and HE counterparts. Due to the large amount of memory required, only S2 cell level 16 and Geohash length 4 to 6 were used for evaluation. The object count of 14,235 takes up half a gigabyte of memory. In the figures that evaluate the number of objects against memory usage and generation times, Geohash and S2 cell sets are used. That's why there are four points. The last one always represents the S2 cell count of 14,235, much higher than the Geohash counts used for the tests.

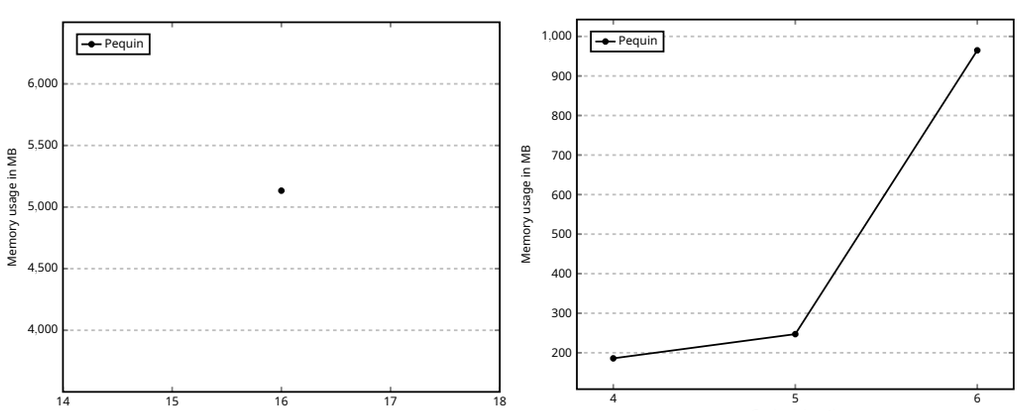


Figure 7.27: Maximum memory consumption in MB for proof generation depicted for S2 cell level 16 (left) and different Geohash lengths (right). The object count (14,235) for S2 cells only produced reasonable results at level 16. The black line represents the Pequin scheme.

The durations for the proof generation in Figures 7.29 and 7.30 vary from 11.55s up to

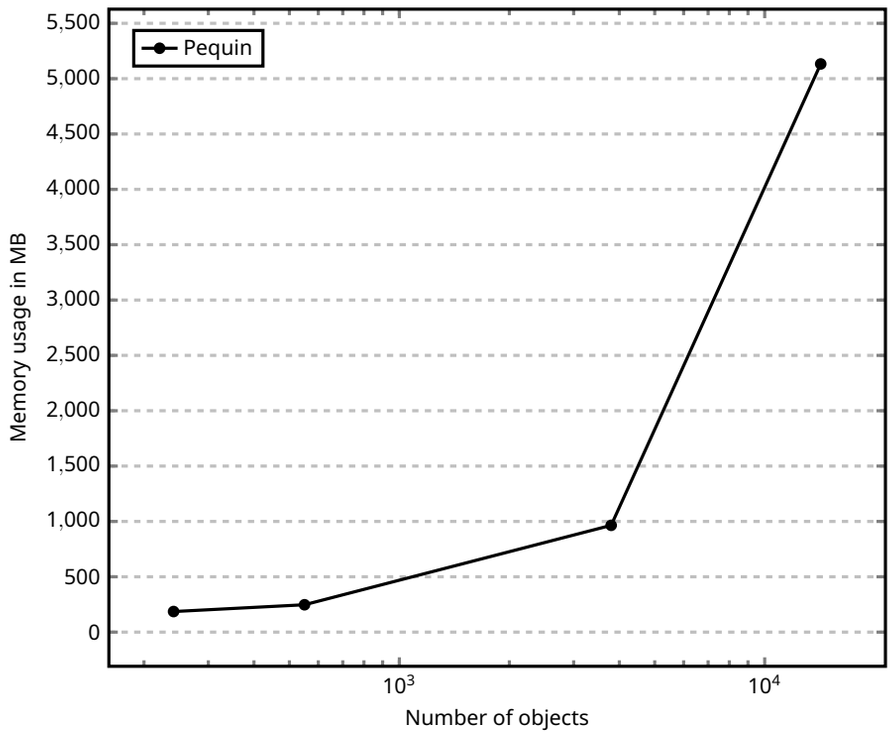


Figure 7.28: Maximum memory consumption in MB for proof generation depicted for different object counts (both S2 cells and Geohashes). The object count (14,235) for S2 cells only produced reasonable results at level 16 (last dot). The other three dots are object counts for Geohashes. The black line represents the Pequin scheme.

267.825s. This is also a limiting factor for use cases that require fast position updates. However, verification times are very fast. All test values are verifying the proof under 0.2s and go down to 0.025s. This is shown in Figures 7.31 and 7.32.

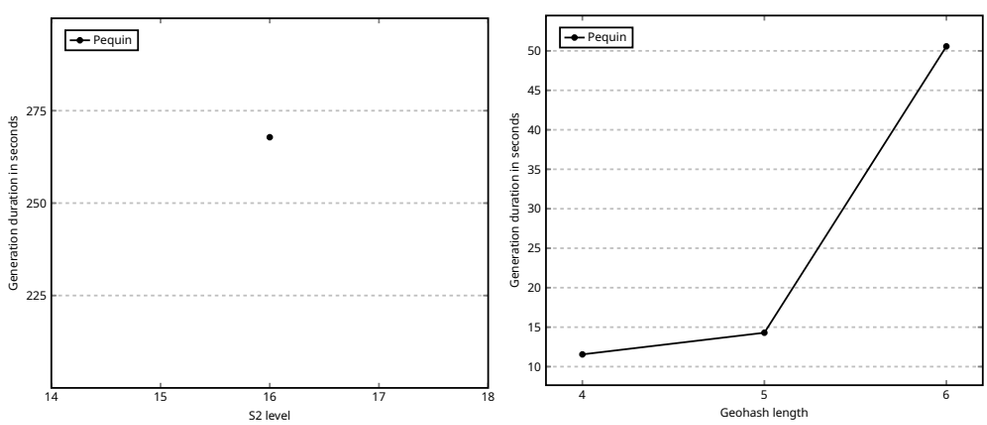


Figure 7.29: Proof generation durations in seconds depicted for S2 cell level 16 (left) and different Geohash lengths (right). The object count (14,235) for S2 cells only produced reasonable results (267s) at level 16. The black line represents the Pequin scheme.

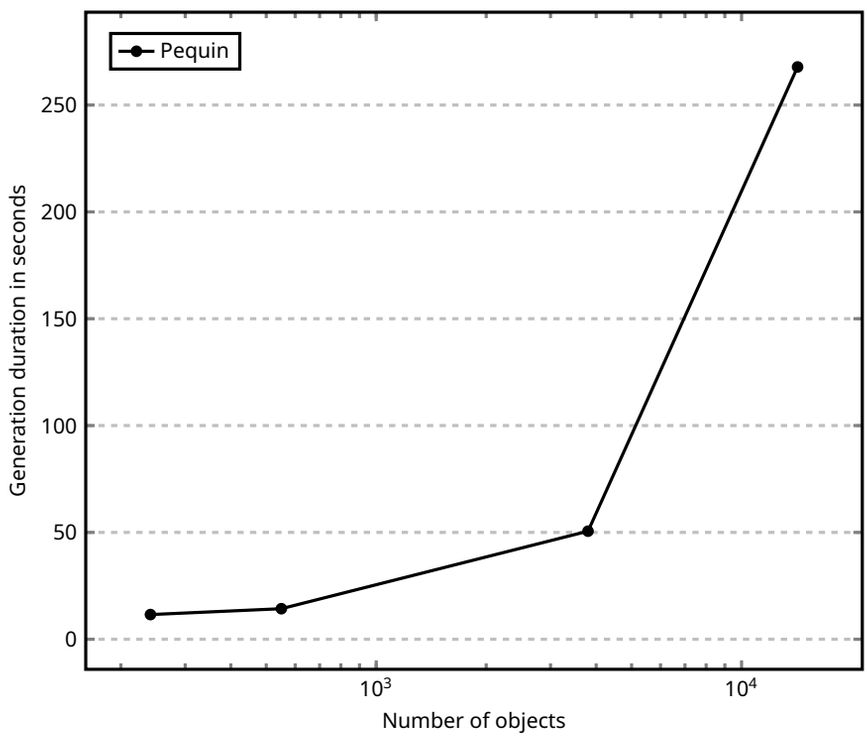


Figure 7.30: Proof generation durations in seconds depicted for different object counts (both S2 cells and Geohashes). The object count (14,235) for S2 cells only produced reasonable results at level 16 (last dot). The other three dots are object counts for Geohashes. The black line represents the Pequin scheme.

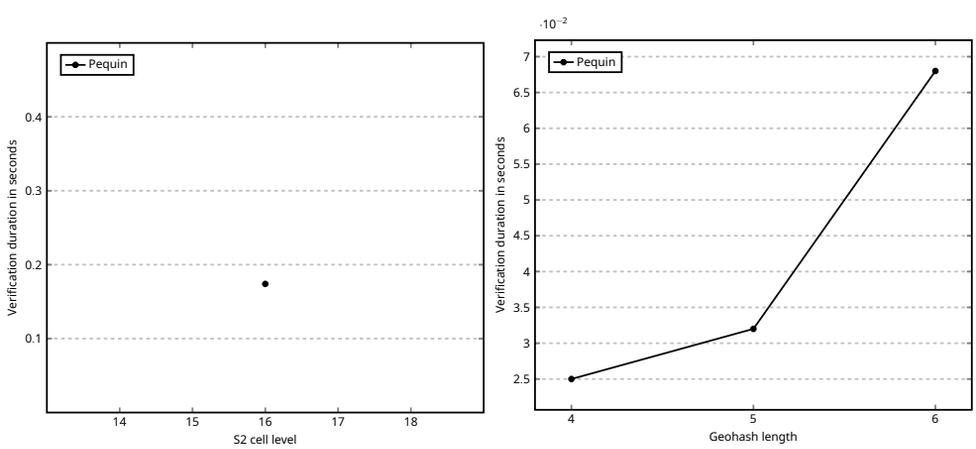


Figure 7.31: Verification durations in seconds depicted for S2 cell level 16 (left) and different Geohash lengths (right). The object count (14,235) for S2 cells only produced reasonable results (267s) at level 16. The black line represents the Pequin scheme.

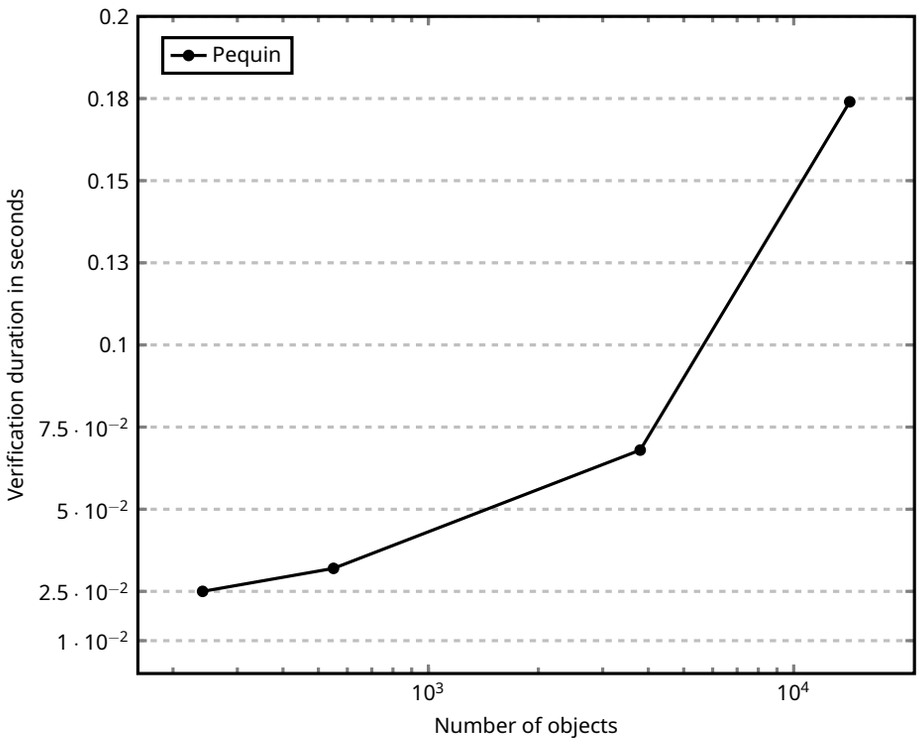


Figure 7.32: Verification durations in seconds depicted for different object counts (both S2 cells and Geohashes). The object count (14,235) for S2 cells only produced reasonable results at level 16 (last dot). The other three dots are object counts for Geohashes. The black line represents the Pequin scheme.

7.3.5.2 Set membership and range-proof library

In this section, the evaluation results of the set membership ZKP scheme introduced in Section 5.3.6.2 by Morais et al. [227] are shown. The library [241] is written in the Go language and can also perform zero-knowledge range proofs. The library is based on the ZKP scheme Bulletproofs [255].

Figures 7.33 to 7.36 show the *setup*, *proof*, and *verification* durations for one proof, i.e., checking whether *one* element is included in a set. The diagrams contain times for different set sizes of 1,000 to 50,000 elements. The setup is linear with the number of elements, while the times for proof generation and verification are constant.

The following diagrams (Figures 7.37 and 7.38) show the *setup*, *proof*, and *verification* durations for 30 proofs, i.e., checking whether one of 30 elements is within the tested sets. This reflects that an S2 point has 30 parents that must be checked against one or more geofences consisting of different sets of S2 cells, i.e., set elements. The figures include durations for different set sizes, 1,000 to 5,000 elements. The results are similar to those with one proof, i.e., linear for the setup and constant for proof and verification.

7.3.5.3 Range Proof for area limitation

The library [241] evaluated in the previous section can be used to check numerical ranges in a zero-knowledge fashion. Figure 7.39 shows results for using the range proof to evaluate whether grid cells are below a certain threshold to address use cases where the service checks whether a client-generated geofence area is within a maximum. For example, the athlete may not reveal her geofence locations in the privacy-preserving whereabouts evaluation. However, the service must limit the number, and the maximum

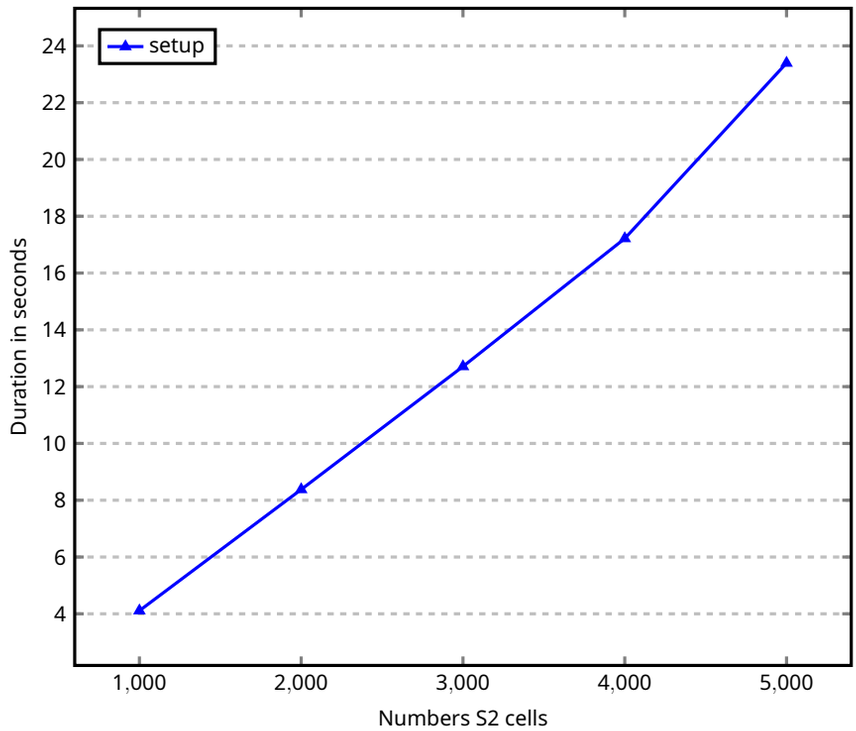


Figure 7.33: Setup duration in seconds of ZKP set membership depended on the number of objects (1,000-5,000). The times are dependent on the number of objects in a linear fashion.

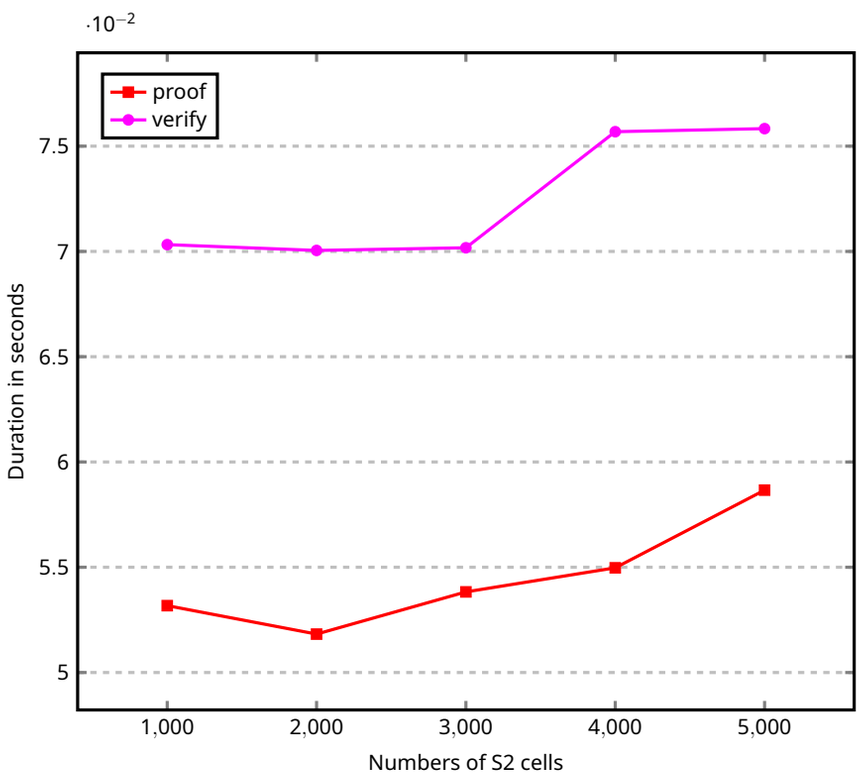


Figure 7.34: Proof generation and verification duration in seconds of ZKP set membership depending on the number of objects (1,000-5,000). The durations are very similar, even for higher numbers of objects.

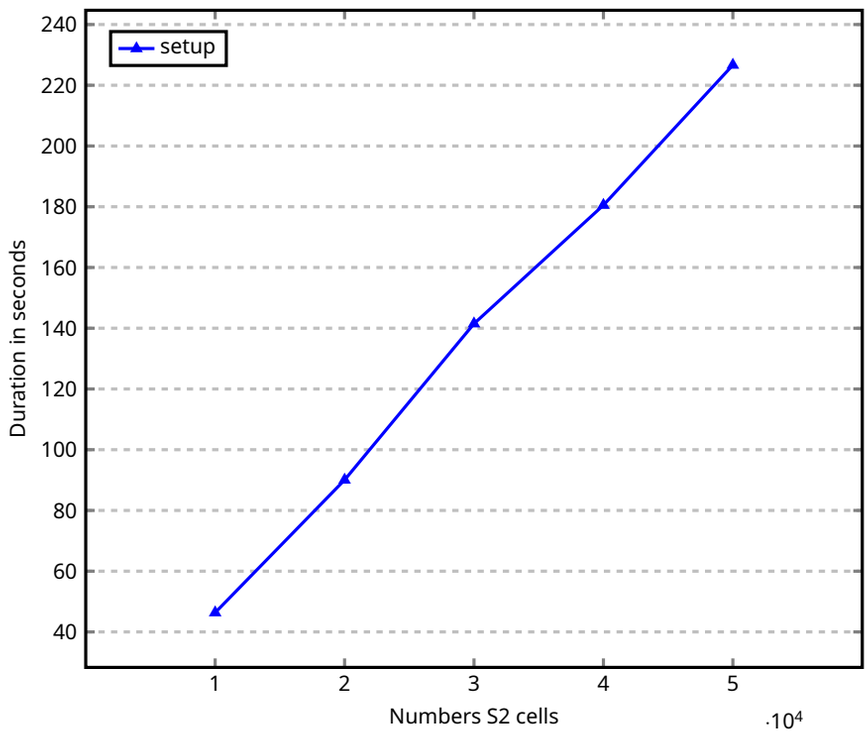


Figure 7.35: Setup duration in seconds of ZKP set membership depended on the number of objects (10,000-50,000). The times are dependent on the number of objects in a linear fashion.

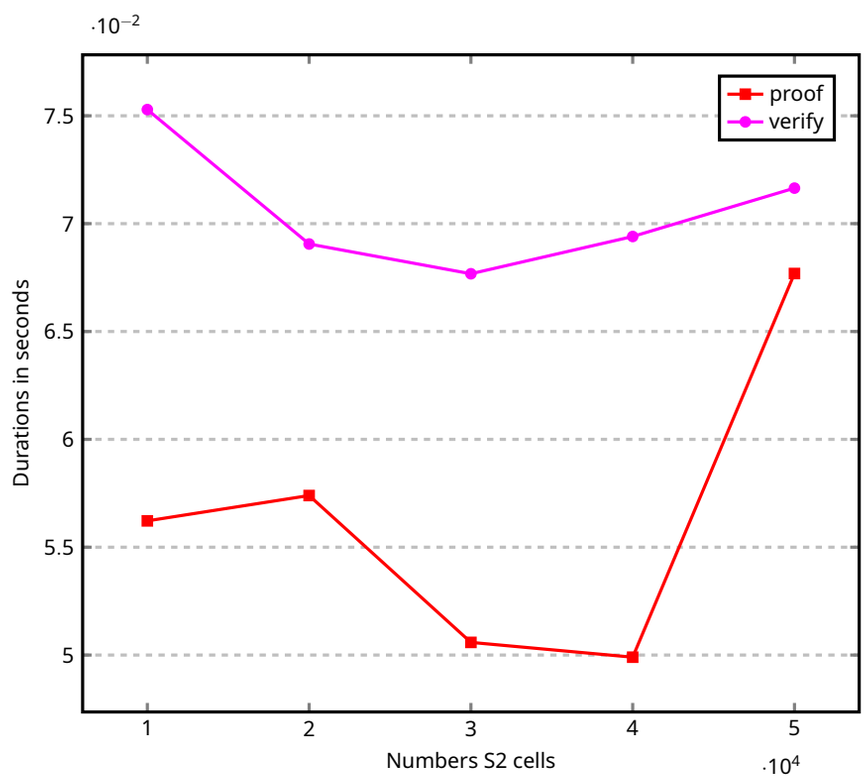


Figure 7.36: Proof generation and verification duration in seconds of ZKP set membership depending on the number of objects (10,000-50,000). The durations are only varying slightly, even for higher numbers of objects.

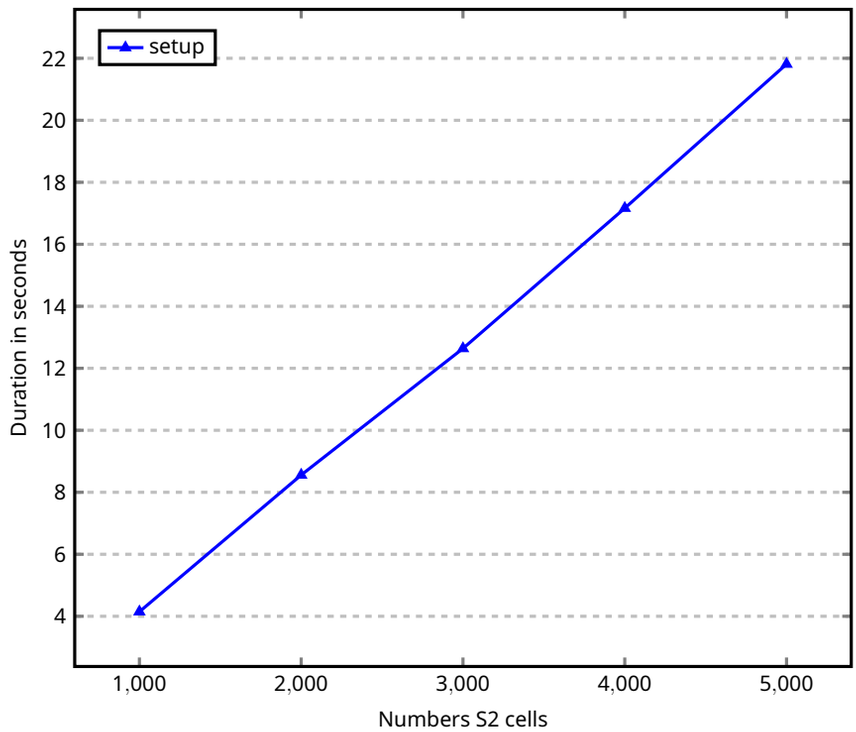


Figure 7.37: Setup duration in seconds of ZKP set membership depended on the number of objects (1,000-5,000) for testing if one of 30 parent values are matching with a set element. The times are dependent on the number of objects in a linear fashion.

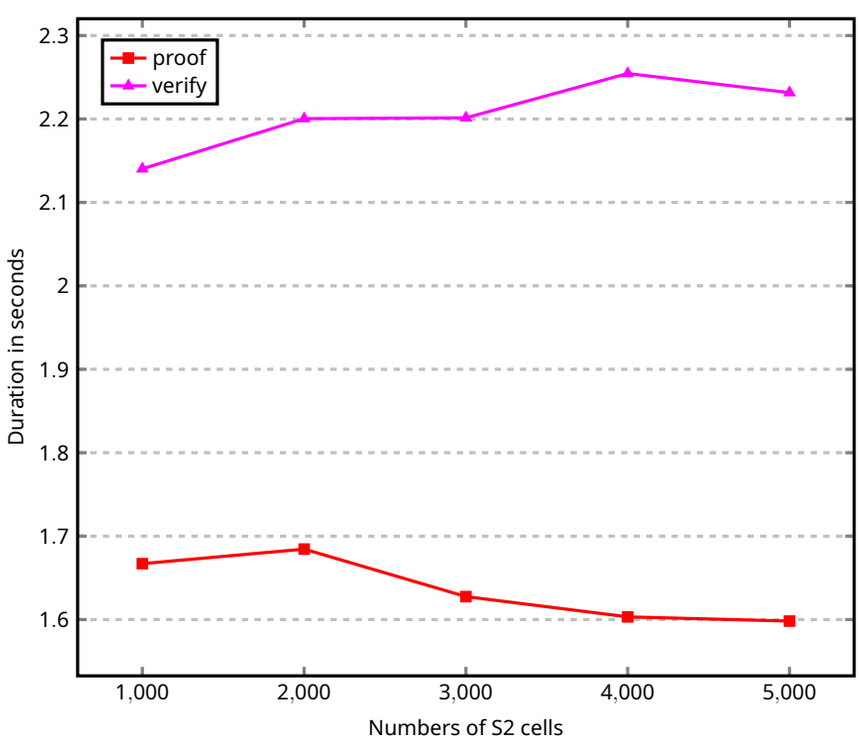


Figure 7.38: Proof generation and verification duration in seconds of ZKP set membership depending on the number of objects (1,000-5,000) for testing if one of 30 parent values are matching with a set element. Even for higher numbers of objects, the durations only vary slightly.

area declared private in this case. The tests show that the setup is independent of the number of cells, but proof generation takes an average of 29.6 seconds per 100 elements, and verification is about 14.9s.

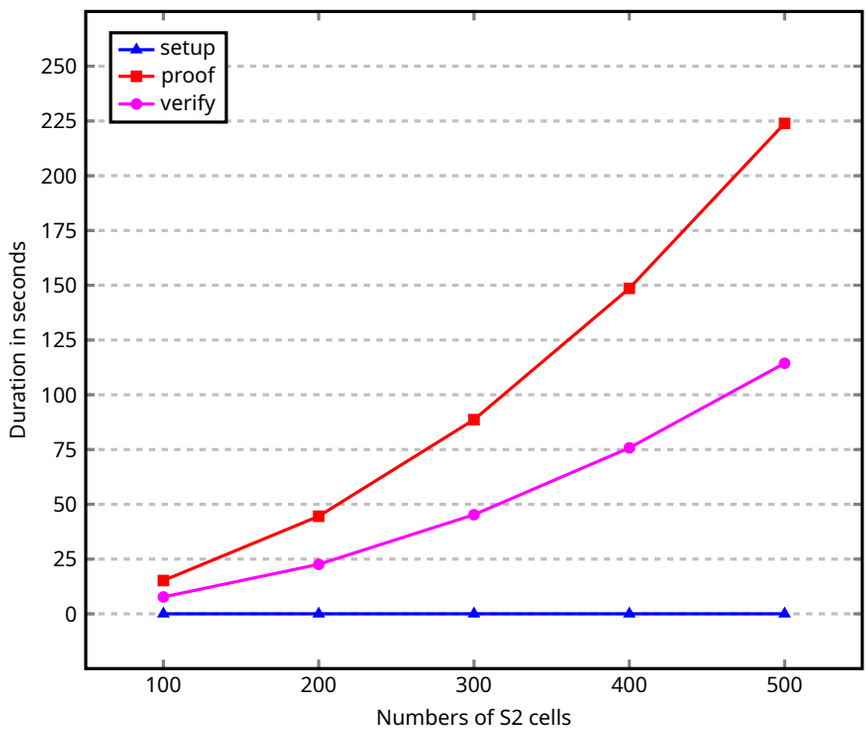


Figure 7.39: Setup, proof generation, and verification durations for different numbers of the range proof algorithm used for evaluating cell sizes in zero knowledge. The blue line indicates the setup durations, which are independent of the number of cells tested. The red line indicates the proof generation durations, which are up to 223 seconds for 500 elements. The magenta line indicates the verification times to check the ranges of elements. Verification times are around half of the time it takes to generate a proof.

7.3.6 Overview table

Table 7.14 summarizes the results of the methods GCH (SHA512), NEXUS, NEXUS-Parallel, and Pequin (ZKP). It allows a direct comparison of storage requirements and durations for three different cell counts (241, 550, and 3,800). Though, it does not include the findings of the Go library [241].

7.3.7 Attribute-based encryption

The evaluation based on the attribute-based encryption protocols is twofold. There is a performance evaluation for the use of ABE with location attributes for different mobile devices. Furthermore, an ABE library comparison focused on the most popular implementations.

7.3.7.1 Location attributes

These results were published in Zickau et al. [190] and are based on an Android app using ABE with location attributes. ABE is typically utilized in a hybrid encryption scheme. AES is applied as the symmetric part, but only the asymmetric part of the decryption process has been measured in the tests.

In the following, an evaluation of the decryption process in dependency on the complexity

Table 7.14: Schemes evaluation overview, GCH, HE, and ZKP

Type	#Cells	GCH	NEXUS	NEXUS-Parallel	Pequin
Loading memory [MB]	241	39.77	20.62	191.14	609.02
	550	41.54	23.43	197.63	871.78
	3,800	42.06	38.71	213.67	2834.55
Loading durations [s]	241	0.015	84.78	17.24	38.89
	550	0.015	186.23	39.75	40.95
	3,800	0.02	1310.02	286.05	140.77
Proof generation [MB]	241	41.02	21.22	194.88	185.90
	550	41.61	23.14	200.62	247.18
	3,800	42.13	40.77	230.72	964.82
Proof generation [s]	241	0.01	2.79	0.91	11.55
	550	0.01	5.97	1.71	14.30
	3,800	0.01	40.76	10.30	50.57
Verification [s]	241	0.006	0.28	0.28	0.03
	550	0.007	3.28	3.27	0.03
	3,800	0.008	26.70	27.93	0.07

of the policy and the key generation is presented. For the tests, the following mobile devices were included.

- Nexus 4 with a Qualcomm Snapdragon S4 Pro APQ8064 at 1.5 GHz
- Nexus 5 with a Qualcomm Snapdragon 800 at 2.26 GHz
- Personal computer running an Intel i5-2410M CPU at 2.30 GHz with 8GB RAM

Each test was run five times, and each data point in the figures indicates the average of these five runs. The performance results for the decryption process, and the key generation can show the application's usability.

Figure 7.40 shows the decryption times with the number of attributes when applying a policy with only logical conjunctions (*and*-operator) of attributes. Figure 7.41 shows the decryption times with the number of attributes when computing a policy with only logical disjunctions (*or*-operator) of attributes. For the policy with only conjunctions, the complexity arises linearly. The mobile devices took almost twice the time for twice the attributes.

For policies with attributes connected by an *or*-operator, the time seems to be constant for the Nexus 4. The varying times for the Nexus 5 suggest that there might still have been background tasks running simultaneously. This result matches the expectation that when only one attribute is required to satisfy the policy, then the length of the policy does not seem to impact the decryption time. The PC is much faster for both results and only takes an insignificant amount of time because it can run the ABE code natively.

As for the key generation, the mobile devices perform almost equally well (Figure 7.42). A good rule of thumb seems to be that the generation takes a second for each attribute in a key. Again, the PC outperforms mobile devices. Overall, the performance of the ABE system seems to be usable for mobile devices. However, outsourcing parts of the decryption should alleviate most of the problems encountered with more attributes.

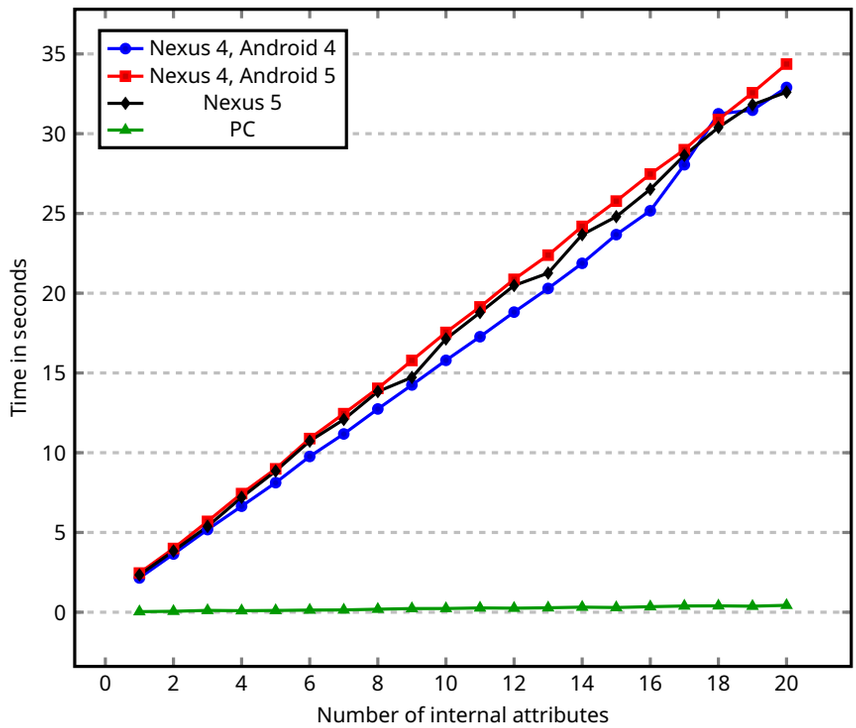


Figure 7.40: ABE decryption performance for location policies with an and-operator.

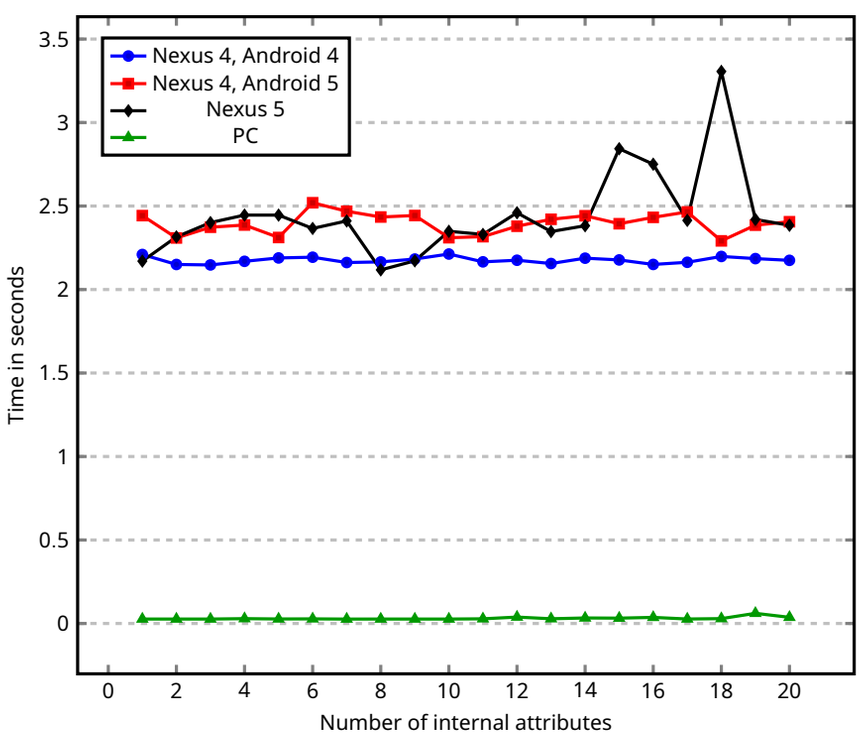


Figure 7.41: ABE decryption performance for location policies with an *or*-operator.

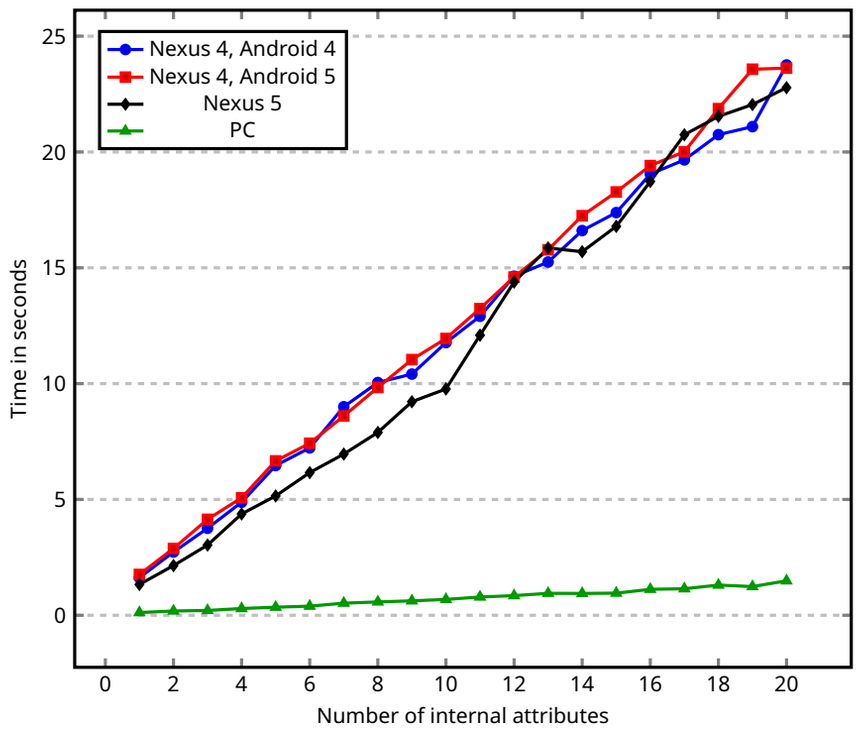


Figure 7.42: ABE key generation performance for location attributes.

Further research will show how many attributes will be necessary in a common use case.

The following additional ABE performance evaluation was previously published by Denisow et al. [197]. To get an impression of how the number of location attributes is influencing the time to decrypt a file, a performance test on a PC with JCPABE [243], see Section 7.3.7.2, was executed. The processor and PC attributes are also as follows.

- Intel CPU i5-2410M at 2.3GHz with 8GB RAM
- Java-Version 8u45

The test run was performed with 2^0 to 2^{10} location attributes. As Figure 7.43 uses a logarithmic representation on the x-axis, the linear results are not apparent at first sight. The different conjunctive location attributes were used in a disjunction. An application scenario for such a large amount of location attributes could be the usage of categories of locations, such as office buildings, public transport stations, university campuses, school grounds, medical facilities, and factory plants. These categories could then be cross-referenced in a contact tracing app for risk contacts of positive-tested people during a pandemic. The goal is to find out where virus-spreading situations occur to adapt and tweak countermeasures.

7.3.7.2 ABE libraries

Part of the evaluation is an overview of features and performance measurements of different ABE implementations. The focus is on ABE schemes with an implementation that could be evaluated. The various schemes are described in terms of *programming language*, *expressiveness*, *performance*, *extensibility*, and *available features*. Since part of the evaluation aims to run the ABE libraries on mobile or low-end devices, such as smartphones and Raspberry Pis, a review of their usability is given. This evaluation was part of a review of the applied ABE schemes survey paper by Zickau et al. [256].

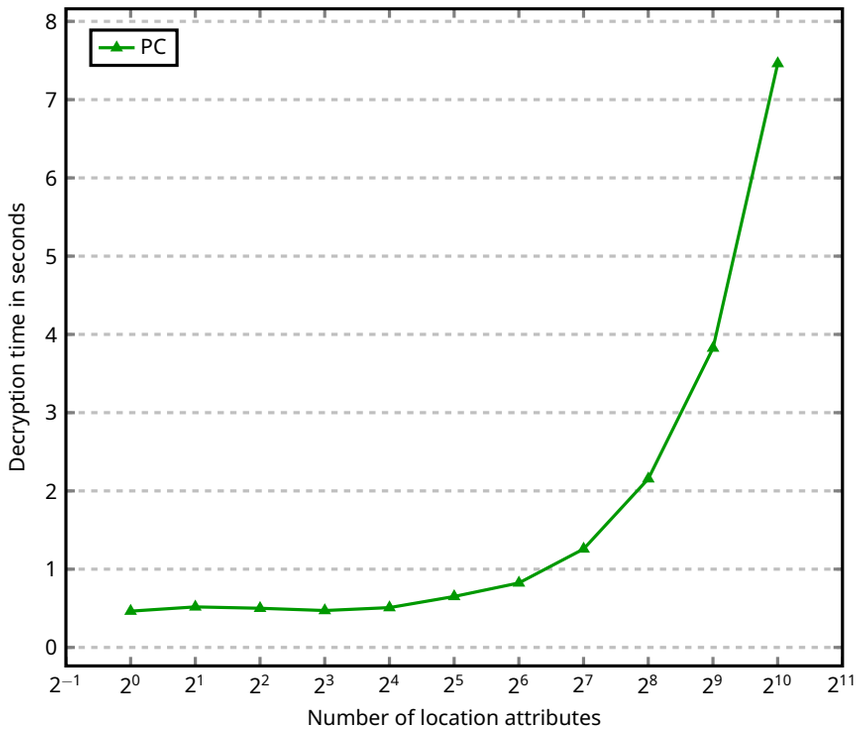


Figure 7.43: ABE decryption times for larger numbers using conjunctive location attributes in a disjunction policy. The x-axis is in logarithmic scale.

Additional advances in the adopted KP-ABE, such as traitor tracing, revocation, and large universes, were implemented by Zhen Liu and Duncan S. Wong in 2014 [257].

cpabe cpabe [192] is the first implementation of a CP-ABE scheme. John Bethencourt wrote it in 2007, a year after the proposed ABE scheme was described. It is based on pairing-based cryptography [258]. It supports only CP-ABE and re-formats the Boolean formulas into an internal format. It also supports numerical attributes and their operators in the policies. In addition to supporting the operators *and* and *or*, it can also handle the *of*-operator. The cpabe library is written in C. Attempts to compile it for an Android device failed due to problems with dependencies. The software is divided into two parts, the libbswabe contains the four basic operations of the scheme, namely *setup*, *keygen*, *encrypt*, and *decrypt*. It can only encrypt and decrypt group elements. The command-line tool for cpabe can perform libbswabe operations on actual data using a hybrid encryption approach with AES-128 in the Cipher Block Chaining (CBC) mode using OpenSSL [259].

libfenc Libfenc was written by Akinyele et al. in 2010 and is a library for functional encryption schemes [260]. It supports both a CP-ABE and a KP-ABE scheme. The library uses the code of the parser of Bethencourt's implementation. Therefore it also supports numerical attributes. libfenc is written in C/C++, making it difficult to run on Android. However, it has been successfully tested on mobile iOS devices [261]. At the time of writing, the project is regarded as abandoned.

Charm Charm [262] is generally a prototyping cryptography system and has been in development since 2011. It provides several ABE scheme implementations, including a reimplementation of the Bethencourt scheme and a KP-ABE scheme. A drawback is the lack of parser support for numerical attributes and the *of*-operator. The library is

based on *elliptic curve cryptography* [90], providing different curves for each scheme. It is written in Python, and a package exists for the Android operating system. Table 7.16 lists four implementations of different schemes of the Charm library, see Row 4 for details. All these schemes, except YCT14, provide the basic ABE constructs of encrypting a random group element. This random element can be hashed to create the key bytes for a symmetric block cipher. This enables the library to use key encapsulation mechanisms and, therefore, deployable in applications. The schemes implemented in Charm can be divided into three groups: *plain* ABE without any additional functionality as shown in Chapters 4 and 5, *multi-authority* schemes, and *proxy re-encryption* schemes. The following paragraphs provide an overview of these schemes and their functionalities.

Charm schemes The schemes BSW07 [192], LSW08 [263], Waters11 [264], YCT14 [265], RW12 [266], OT12 [267], LW12 [268], and YAHK14 [269] implement the four basic methods, namely *setup*, *keygen*, *encrypt*, and *decrypt*. Since the revocation and negation of attributes are not compatible with the superclass of the standard *ABEEnc*, the library LSW08 does not fully implement the definition referenced in [268] is based on. The YCT14 is the fastest scheme among the Charm libraries because it is the only scheme not based on pairing, given the Diffie-Hellman assumption over elliptic curves (ECDDH). RW12 and OT12 implement the CP-ABE and KP-ABE constructions. LW12 describes a different CP-ABE construction, the KP-ABE model. YAHK14 provides a non-monotonic and unrestricted CP-ABE scheme, in which attributes are represented as integers in $\mathbb{Z}_p$.

The schemes in LW11 [270], RW15 [271], YJ13 [272], and YJ14 [273] provide a multi-authority setting. Using the technique of updating the secret attribute component for non-revoked users, the implementations YJ13 and YJ14 provide user attribute revocation. LW11 and RW15 are the only schemes that do not introduce a single authority but rather implement an autonomous one. In this case, both implementations rely on global public parameters.

The implementations LWW14 [274] and YJ13 [272] provide proxy re-encryption for different use cases. The proxy in LWW14 converts the ciphertext for time and locks the description to (multiple) time ranges with a granularity of one day. Only users with the appropriate time-based attributes can decrypt the ciphertext message encrypted by the proxy. In YJ13, the proxy re-encrypts the ciphertext and relieves the client from performing the pairing operation. The client only needs to perform exponentiations to recover the ABE-ciphertext.

The predicate-based encryption library (PEBEL) [275] builds upon Charm. It extends the functionality presented in Charm by implementing a hybrid encryption scheme, with ABE used as a key encapsulation scheme. It also implements *numerical* attributes and the *accompanying* operators, but it does not implement the *of* operator.

The following section overviews the benchmark tests evaluated for the given Charm schemes.

Charm implementation benchmarks The schemes described above are very different in their functionality. This is addressed in the benchmark results in Figure 7.44, Figure 7.45, and Table 7.15. The basic operations, *setup*, *keygen*, *encrypt*, *decrypt*, are provided by all libraries. In addition to these functions, the operations *authority setup* (authsetup) and *re-encryption* (re-encrypt) are evaluated. The results show a cumulative execution in

seconds of 100 iterations of a full cycle for each scheme.

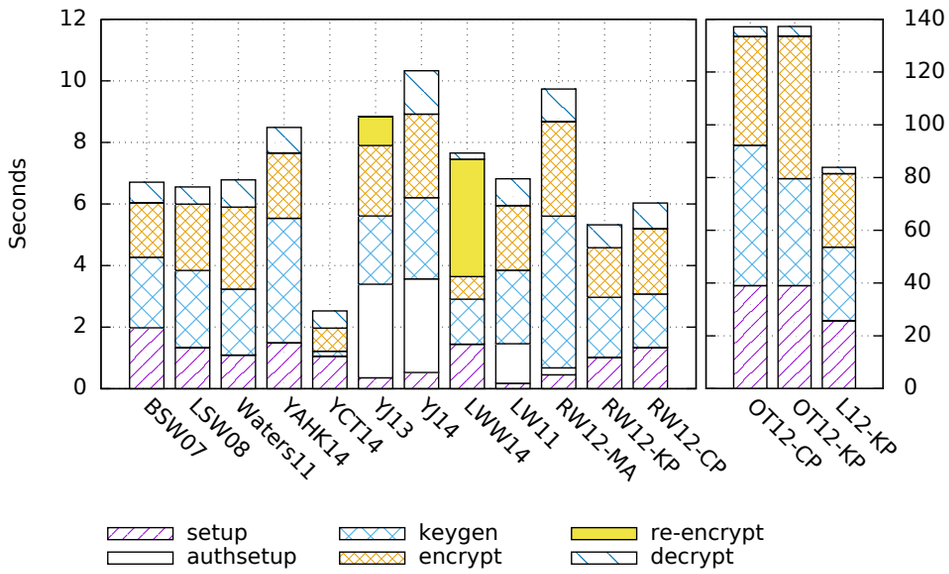


Figure 7.44: Charm Benchmark on a i7-4710HQ @ 3.40 GHz (cumulative time for 100 iterations)

Table 7.15 shows the results for Figure 7.44. The policy used for the benchmark test is simple: *THREE and (ONE or TWO)*. The authsetup is necessary for multi-authority schemes and can be considered part of the setup phase, which generally has only a single authority. Examples of that are the schemes BSW07 and LW11.

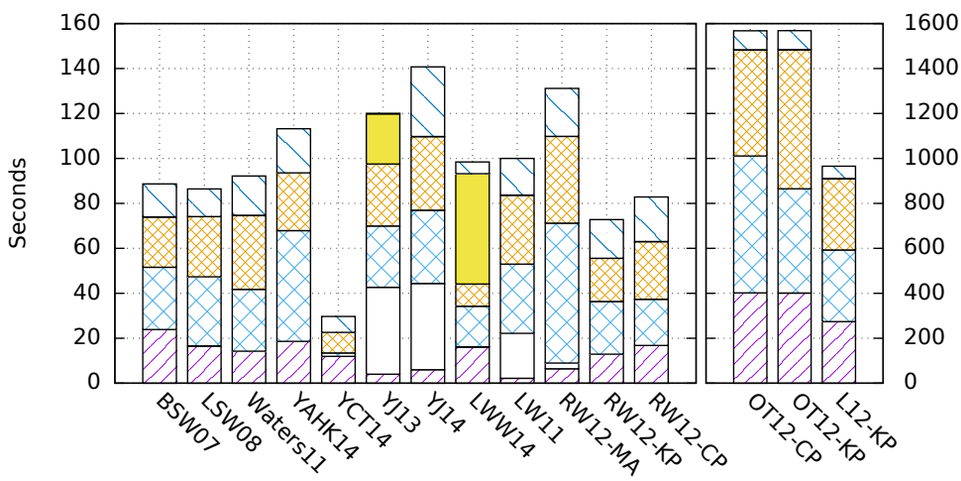


Figure 7.45: Charm Benchmark on a Raspberry Pi 2 @ 900 MHz (cumulative 100 iterations) in seconds

For a scheme, such as non-monotonic YAHK14, the setup needs to generate more components during *keygen* because the negation of attributes needs to be represented. Compared to single authority or proxy-less schemes, this phase takes almost twice as much time as the other schemes. However, this is still more efficient than the naive approach of adding each negated attribute as a single attribute to the user's secret key.

As mentioned before, YCT14 does not require bilinear pairing, the slowest operation overall, and is, therefore, the fasted scheme. In YJ13, the ciphertexts are encrypted based on the ElGamal scheme, see Appendix A.2.1, with the decryption based on only a single exponentiation and is therefore also a high-speed scheme compared to the other methods based on different cryptographic protocols.

Dual Pairing Vector Spaces (DPVS) is a technique for simulating orthogonal subgroups occurring in composite order groups as the basis for the ABE implementations in LW12 and OT12. The three implementations that use DPVS are about ten times slower than those based on prime-order groups.

Additional Charm details All ABE Charm implementations are based on bilinear pairing, defined over elliptic curves. Charm supports three pairing backends, Pairing-based Cryptography (PBC) [276] as the default, the MIRACL [277] cryptography library, and the RELIC [278] cryptography meta-toolkit. Each of these libraries instantiates its own elliptic curves. Some are type A symmetric pairing over the SS512 supersingular curve or Type D asymmetric pairing MNT224. LW11, LWW14, YJ13, and YJ14 support only symmetric pairings.

The security proof of LW11 is only available for composite order groups instead of prime order. Since Charm does not provide access to composite order groups, the scheme was implemented over prime order groups. This should be considered insecure and prototypical because of the unknown security-bound implications.

Table 7.15: Charm Benchmark on a i7-4710HQ @ 3.40GHz (cumulative time for 100 iterations)

Scheme	Setup	Auth-Setup	Keygen	Enc	ReEnc	Dec
BSW07	1.979	0.000	2.289	1.775	0.000	0.661
LSW08	1.338	0.000	2.501	2.154	0.000	0.559
Waters11	1.092	0.000	2.141	2.665	0.000	0.885
YAHK14	1.496	0.000	4.037	2.123	0.000	0.830
YCT14	1.056	0.000	0.161	0.749	0.000	0.557
YJ13	0.355	3.038	2.217	2.286	0.934	0.014
YJ14	0.526	3.037	2.638	2.716	0.000	1.408
LWW14	1.447	0.000	1.458	0.733	3.813	0.200
LW11	0.177	1.283	2.386	2.103	0.000	0.868
RW12-MA	0.450	0.227	4.929	3.072	0.000	1.053
RW12-KP	1.023	0.000	1.947	1.608	0.000	0.741
RW12-CP	1.340	0.000	1.733	2.130	0.000	0.828
OT12-CP	36.141	0.000	49.184	38.233	0.000	3.324
OT12-KP	36.116	0.000	37.522	50.019	0.000	3.322
L12-KP	23.854	0.000	25.754	25.772	0.000	2.193

cpabe (Java) A Java version of cpabe was written by Junwei Wang [279] in 2013. It is a direct port of John Bethencourt's implementation of cpabe. This became apparent because it is based on the same internal structure and rarely takes advantage of Java's object-oriented paradigm. This implementation uses the same internal format of policies that the C implementation introduced. Since there is no policy parser, there is no way to enter policies conveniently. Instead, the user must manually enter the reverse polish notation of the intended policy tree [279]. The only library dependency is the JPBC [280], which is required for the cryptographic primitives within the ABE scheme. It is possible to run cpabe (Java) on any machine running Java, including Android.

JCPABE(2) JCPABE(2) is based on cpabe (Java). It complements and improves the existing functionality. A policy parser has been added to enter policies and numerical attributes the same way as in the Bethencourt version of cpabe. The parser was written using JavaCC and JJTree. Since JCPABE2 is based on cpabe (Java), it inherits the platform independence. No new dependencies have been added. The library has been tested to run on Android devices. The extensions to the functionality are described in Denisow et al. [197] and Section 6.2. They include the generation and appending of attributes to an existing private key and an experimental feature that allows for *dynamic location* information within policies, see Section 7.3.7.1.

jTR-ABE jTR-ABE [281] is based on an early version of JCPABE. Like JCPABE, it supports expressive policies with numerical attributes and threshold gates. The implemented LW14 scheme enables public black box traitor tracing. It is possible to identify the users who have shared their private keys to create a decryption black box using special iterated encryption and decryption challenges that can be executed publicly and outside of the trusted central authority. The maximum number of unique users throughout the system's lifetime must be defined in the setup phase for this scheme.

KPABE Yao Zheng implemented [282] the GPSW06 [191] scheme, which was the first published KP-ABE scheme, and applied it to personal health records (PHR) for privacy-enhancing [283]. The implementation is derived from the Bethencourt implementation of CP-ABE [192]. It is also divided into the ABE library and a set of command-line tools. The policy syntax supports conjunctions, disjunctions, and general threshold gates but lacks support for numerical attributes.

DCPABE Stefano Braghin implemented the decentralized CP-ABE scheme LW11 [270] [284]. It is based on a linear secret sharing scheme (LSSS) for applying the policy application and supports conjunctions and disjunctions. Numerical attributes are not available.

DET-ABE Miguel Morales-Sandoval implemented DET-ABE [285], another realization of the BSW07 scheme [192]. It supports three security levels by selecting appropriate AES key sizes (128-, 192- or 256-bit) and elliptic curves for bilinear pairings (Type A of 256-bit size or Type F of 384- / 512-bit sizes). The library currently supports threshold policies in postfix notation without explicit *and-or*-operators or numerical attributes. The class model of the library represents a framework since the implementation of the ABE scheme is encapsulated in various components such as TrustedAuthority or DETABECipher. These components cannot be used directly because they lack the standalone functionality as a server, e.g., servlet, or client, e.g., command-line tool.

PIRATTE In [193], Jahid et al. introduces a proxy-based immediate revocation scheme and implementation [286] based on Bethencourt's CP-ABE scheme and source code. The minimally trusted PIRATTE Proxy handles user and attribute revocation. In a case study, PIRATTE is the underlying cryptography scheme in EASiER, an access control architecture for Online Social Networks (OSN), and in DECENT, a decentralized architecture for OSNs. The EASiER implementation consists of a Facebook application demonstrated by ABE.

arcanum Arcanum [287], the successor of JPBC, is a Java library that implements four ABE schemes that are based on Garbled or Arithmetic Circuits built on top of Lattices or Multilinear Maps: GVW13 [288], GGHVV13 [289], GGHSW13 [290], BNS13 [291]. GVW13 [288] additionally utilizes a Two-to-One Recoding scheme. The library is built modularly and supports different primitives.

LSSS2 The LSSS2 project [292] implements the large universe construction of Waters11 [264]. The related article [293] describes efficient pairing algorithms and other optimizations for multiplication and exponentiation. It is based on the ate-pairing library, which applies the pairing-friendly Barreto-Naehrig curves. The software package requires Xbyak5 – a just-in-time (JIT) assembler – and it is only supported only on 64-bit systems. LSSS2 supports only group element encryption but could be extended to encrypt arbitrary data with a hybrid encryption approach.

NEON ABE NEON is a single instruction multiple data (SIMD) set for ARM processors that can be used to implement a fast ABE scheme. Sánchez et al. [294] have proposed such an implementation based on the CP-ABE scheme of Waters11 [264]. The implementation is very rudimentary, i.e., it does not provide policy parsing or even secret sharing. It can be built for NEON-based chips, but the software package does not include build instructions. With a few tweaks on a Raspberry Pi 2 with the ARMv7 architecture, executing the included benchmark code, which provided the NEON instruction set, was successful. A sample implementation of the complete encryption/decryption cycle is missing, making it impossible to verify whether the ABE scheme is implemented correctly.

AndrABEn Ambrosin et al. have developed the library AndrABEn [295] specifically for Android devices [296]. It is based on the C libraries cpabe and KPABE and implements the schemes BSW07 and GPSW07. Since it contains libbswabe, it should be licensed under the GPLv2. The related paper focuses on the performance of the implementation tested on Android devices.

Table 7.16: ABE implementation libraries overview

Library / Impl.	Impl. Author(s)	Scheme(s)	Language	Based on Lib.	License	Updated
cpabe	John Bethencourt	BSW07	C	PBC	GPLv2	03/2011
libfenc	Matthew Green, Joseph Ayo Akinyele	LSW08, Waters11, GPSW07	C++	PBC	GPLv2	02/2011
Charm	Joseph Ayo Akinyele	BSW07, LSW08, Waters11	Python	PBC, MIRACL or RELIC	LGPLv3	07/2013
	Gary Belvin	LW11			LGPLv3	11/2013
	Artjom Butyrtschik	LWW14, YCT14, YJ13, YJ14b			LGPLv3	07/2015
	Alexander Förster	YAHK14			LGPLv3	07/2015
	Yannis Rouselakis	RW15, LW12 (KP), OT12 (KP+CP), RW12 (KP+CP)			Unlicensed	11/2012
cpabe (Java)	Junwei Wang	BSW07	Java	JPBC	GPLv2	03/2015
JCPABE	Iwailo Denisow	BSW07	Java	JPBC	GPLv2	06/2015
jTR-ABE	Artjom Butyrtschik	LW14	Java	JPBC	GPLv2	09/2015
KPABE	Yao Zheng	GPSW07	C	PBC	GPLv2	11/2014
DCPABE	Stefano Braghin	LW11	Java	JPBC	Unlicensed	11/2012
DET-ABE	Miguel Morales-Sandoval	BSW07	Java	JPBC	Public Domain	06/2015
PIRATTE	Sonia Jahid	JB12	C	PBC	GPLv2	04/2013
arcanum	Angelo De Caro	GVW13, GGHVV13, GGHSW13, BNS13	Java	arcanum-pbc	LGPL v3	04/2015
LSSS2	Eric Zavattoni et al.	Waters11	C++	ate-pairing	3-clause BSD	10/2013
NEON ABE	Ana Helena Sánchez	Waters11	C++	-	Public domain	02/2013
AndrABEn	Moreno Ambrosin, Mauro Conti, Tooska Dargahi	BSW07, GPSW07	Java	PBC	GPLv2	10/2014

7.4 Policy-based approach

Based on the use case *Whereabouts*, a policy-based approach is described in Section 5.3.9. Implementation details and screenshots of the user interface can be found in Section 6.3. In addition to the thesis' findings, the following evaluation results are also based on the project publications by Elmasllar and Plass [297], Putschli [298], Herber et al. [37], Schanzenbach and Zickau [39], and Plass and Zickau [230]. Four different stakeholders use the policy-based interfaces: the athletes, the staff of a national anti-doping organization (NADO), the doping control officers (DCO), and their service providers. Besides the discussed privacy aspects of an alternative whereabouts system, the focus was also on usability, user experience, and reliability of the involved stakeholders.

7.4.1 Whereabouts interface

Field tests were conducted to evaluate the PARADISE/EVES system against the then-established ADAMS software during the project. An EVES device was used by an individual acting as an athlete. A total of five DCOs employed by the associated partner Professional Worldwide Controls (PWC) tested the PARADISE system's prototype interface(s). Each DCO test was conducted separately and began with an introduction and open-ended questions [297]. Throughout the project, requirements analyses were based on interviews with multiple athletes, NADOs, and DCOs. A tablet PC was used to access the interfaces. The goal of a DCO was to find an athlete in a real-world scenario. Input from the project partners conducting the tests was kept to a minimum. The two most important results are

- DCOs were not satisfied with the ADAMS system, and
- asked for an additional localization add-on that provides additional information.

During the testing, it became clear that the DCOs were very accustomed to using the information provided by ADAMS, i.e., having access to all of an athlete's calendar entries. DCOs would use past information to estimate an athlete's whereabouts or to assess the likelihood of making contact with the subject. If a potential test attempt were deemed unlikely, a DCO would decide to postpone it and set a *safe hour*. A *safe hour* would be to test athletes early in the morning at their residence. On the one hand, a locating device opens the window for a broad 24h timeframe for testing. On the other hand, the information that an athlete is in a private area can discourage a DCO's attempt. Not having an address or name but having GPS coordinates with a radius of accuracy required the DCO to call the athlete on a cell phone. In testing, DCOs found the athlete within two minutes of the call. That is a concise time frame. WADA's best practice exceptionally allows a one-hour window after a DCO contacts someone by phone to avoid giving the selected athlete any warning. Since the DCOs were only working with an average accuracy radius of 250 meters during testing, they were required to call. This leads to a tradeoff between short warning times and fewer phone calls. A tracking device may lead to situations where a DCO is forced to chase an athlete who is on the move. This would mean more freedom from the athletes' perspective, as they would not be forced to change ADAMS calendar entries on-the-fly. From the DCO's perspective, high-probability test attempts would likely occur less frequently. Several approaches to overcome this problem were discussed during the project, including those presented in this thesis. Areas where an athlete is unlikely to spend much time or that are outside of a typical daily routine, such as airports,

train stations, bus stations, or shopping malls, could be marked. Also, if an athlete is on a train or bus, the information could be displayed to the DCO after requesting a location update again. Otherwise, stored addresses used regularly, such as home, training site, or work addresses, could be displayed by the system if the current location is associated with them. A calendar-based and wearable-supported system could also be a transitional step by using more granular address information, such as city names and zip postal code areas. In most cases, DCOs will test athletes within their city or larger zip postal code areas. In [297], potential risks and disruptions of a system such as PARADISE/EVES are listed.

7.4.2 Data protection goals

The German Standard Data Protection Model (SDM) defines seven goals that serve to systematize the requirements of the GDPR [299]. These goals are *data minimization, availability, integrity, confidentiality, unlinkability, transparency,* and *intervenability*. Herber et al. [37] compare the ADAMS system with the PARADISE/EVES project approaches. The research project results address the goal of data minimization by limiting the amount and storage time of whereabouts data needed. The EVES device ensures that whereabouts data is available to the DCO even if calendar entries are not current or missing. Using Sealed Cloud Technology [300] as the backend environment and logging mechanisms for accesses, the PARADISE system's integrity, transparency, and intervenability are advantageous to ADAMS. The secure backend system ensures confidentiality.

Additionally, the introduction of privacy geofences allows athletes to cloak confidential locales. The same is valid for enabling unlinkability of location information by introducing personal and general geofences and evaluating the hierarchical link between the DCO client and the target. With the availability of the PARADISE approach, it can be assumed that the always-existing encroachment of fundamental rights by the doping control system wanted and needed for sports is disproportionate due to the lack of necessity of a control system without the targeted use of privacy-friendly technologies [301] [37].

Already in 2014, before the project was launched, the sprinter Ruth Spelmeyer from VfL Oldenburg expressed her opinion in an interview [302] when asked about an advanced alternative system.

> *It would make controls easier as you would not be not found. Besides, you are less bounded, and you do not have a fear of forgetting to make an entry in the calendar.*

It can be considered that the research project PARADISE evaluated the possibilities of a positioning wearable and introduced methods and protocols on how to address relevant privacy concerns. As such, a research environment always has to deal with limitations. It is advised to investigate further privacy-preserving mechanisms in international doping controls and athletes' whereabouts.

7.4.3 Semantic hierarchy tests

From the implementation details in Section 6.3.4, the following system-related findings emerge. These are based on the conclusions shown in [38].

Several shortcomings were identified during the testing of OpenStreetMap's Nominatim

service [245] as introduced in Chapter 6. The service only allows one request per second with unreliable response times. The privacy area system uses caching mechanisms to store previous queries in its database. Since the number of results is set to a maximum of 50, it was decided to preload general privacy areas, such as hospitals and churches, into the system to make on-the-fly queries obsolete. For smaller general privacy areas, such as a doctor's office or smaller churches, the minimum radius for the first demonstrator was set at 200 meters. This approach intends that a location that might be detected near a private area would not necessarily reveal its underlying semantics.

During development, we mainly relied on the Service-centric Networking's CATLES [303] [148] service for system testing and other location mocking apps available for Android. CATLES allows simulating the movement of the DCOs as they gradually approach the athletes. In addition to the real-world testing mentioned in the previous section, additional validations were conducted. Project members acted as DCOs and athletes while moving around in Berlin and Potsdam (Germany).

One of the main problems was the reliability and accuracy of the data provided by the Nominatim geocoding service. The results were often *hit and miss*. Sometimes a geofence was returned, while another test of the same scenario returned no geofence. In some cases, no reliable pattern was found. The accuracy of Nominatim's responses also varied depending on the level of the geofence hierarchy. The original intention was to go further down the hierarchy and obtain more detailed semantic geofence, such as buildings and roads. However, the OpenStreetMap data on which Nominatim is built was insufficient, e.g., in a test where the athlete was in the Tiergarten district of Berlin, a query for the street the athlete was on only returned a street of a large, centrally located traffic circle. The same problems occurred in other parts of the city. Sometimes a usable response was returned for a road, and sometimes only a tiny geofence extract revealed the exact location of the athlete. A similar problem occurred when querying buildings where the returned results were incomplete.

Therefore, the levels of the geofence hierarchy were limited. The most minor level used is a city district, as this is the smallest level that provides reliable results (as shown in Figure 5.23). If a DCO is closer, i.e., within the same city borough, a circular geofence was used to favor the initially planned semantic geofence. The radius of the circular geofence used is equal to half of the distance between the DCO and the athlete. The exact location is obscured within this circle by spatial randomization of the athlete's actual position. The circular geofences were also used when Nominatim did not return a geofence. By using circular geofences as a fallback mechanism, reliable results were observed.

7.5 Geofencing on distributed ledgers

As mentioned previously and introduced in detail in Section 5.4.5, the approaches of grid cell geofences and set membership queries were also analyzed in the context of LBS use cases utilizing distributed ledger technology such as smart contracts. The geofences, in this particular case, cover cell tower areas in contrast to the postcode areas covered in Section 7.3.1. This is reflected in the additional encoding evaluation below. The following results were presented in a publication by Victor and Zickau in 2019 [121].

7.5.1 Cell sizes and positioning methods

Using network operator cells as the positioning method within the carsharing use case utilizing smart contracts is proposed. The resolution link between the two location representations is required to determine the maximum level for this use case. This will also determine the maximum cell size within the geofence representation, depending on how accurate a geofence can be to its actual world area compared to the positioning method used in the mobile device.

It must be emphasized that the better the positioning method is, the more accurate geofence representation can be used. The left side of Figure 7.46 outlines a simplified network cell map (honeycomb shapes). This black network cell is covered by a grid of two cells within the geofence representation on the right side. Network operators generally do not disclose cell tower locations and coverage areas. An estimation was made using crowdsourced data [304]. Using crowdsourced data for LBS analytics was previously discussed by Uzun et al. [305] [306].

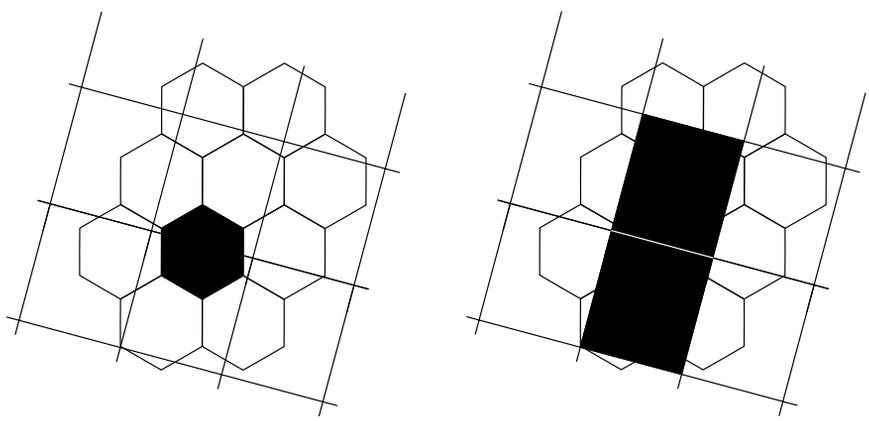


Figure 7.46: A simplified mobile network cell (hexagon grid cells, left), is encoded to be fully contained within two geofence grid cells (quadrilateral grid, right).[4]

7.5.2 City encoding

As in the geo encoding evaluation at the beginning of this chapter, the cell encoding of Germany's seven largest cities by population is used. The encoding was limited to mobile cells for positioning targets with a maximum Geohash length of 5 and S2 cell level 11 to maintain compatibility with the mobile network cell area encoding. The Geohash encoding achieves a similar coverage to S2 encodings but requires more grid cells. Due to the limitation of the minimum grid cell size constraint, no encoding can acquire a coverage below 150%. For example, using a maximum S2 cell level of 11 to cover Berlin leads to the best result of 160% coverage when using 30 S2 cells.

7.5.3 Cell tower encoding

The cell tower coverage areas model was created using an Android application [304] that collects cell tower information and resolves it using APIs from the Google location

[4]Image sources: Figures used in this subsection are from the original publication, Victor and Zickau (2018), and were created by Friedhelm Victor.

services. Mobile device positions consist of a latitude, a longitude, and a horizontal accuracy, as defined on the Android Developer website [307]. A confidence value of 23% means that the target is within the shown circle with a likelihood of 23%. It should be emphasized that the actual shape of a cell is not a perfect circle, but the accuracy value is in the same order of magnitude.

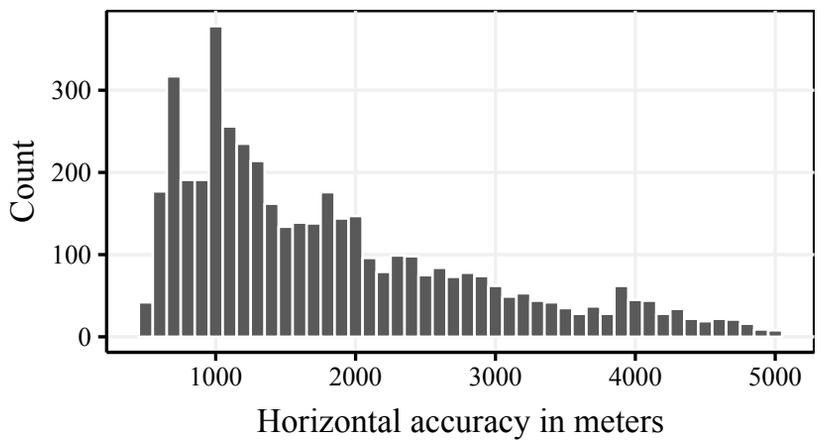


Figure 7.47: Distribution of horizontal accuracy values (radius) obtained for cell tower position estimates from the Google location API.

For this evaluation, 4.505 cell tower position estimates with radii between 500 and 5.000 meters were conducted in Germany. The distribution is shown in Figure 7.47. The majority of the cells have a horizontal accuracy of 500 to 2.000 meters.

All cell tower coverage areas were encoded with S2 cells and Geohashes, with a range of minimum and maximum lengths and levels allowed. For each outcome, it was counted how many grid cells were needed to represent the given polygon. As already explained in Section 7.3.1, the coverage was measured by describing how much the original polygon area was covered. For example, coverage of 130% means that an additional 30% of the excess area that was not part of the original polygon was covered. The best possible value would be 100%. The number of cells required decreases by relaxing the coverage constraint, e.g., by accepting a coverage of < 400%. In all constraint scenarios, the S2 encodings perform better than Geohashes.

For coverage constraints <150% and <200%, the maximum grid cell areas (S2 level 11, Geohash length 5) correspond to approximately area sizes of 15 km^2 to 25 km^2. This determines the choice for the smallest grid cell size used for the geofences since a larger oracle-equipped cell cannot lie within a smaller geofence cell.

Figure 7.48 shows a vehicle trace obtained from simplified mobile network cell positions. There is a gap in the middle of the map that is not due to missing information in this scenario. If a periodic location update was chosen instead, the car might have passed several network cells before the corresponding smart contract is informed about a new position.

Figure 7.48: An example car trace of mobile network position estimates (left), converted to S2 grid cells (right).

7.5.4 Geofence in decentralized ledger technologies

The goal is to encode polygons with as few grid cells as possible, resulting in financial storage and transaction costs for the Ethereum VM (EVM) [62]. Only the initial setup can be financially costly. Data access, which occurs after the setup phase, is comparatively cheap. The goal is to identify the optimal grid cell sizes to configure a smart contract accordingly. Second, the actual execution costs can be measured based on an implementation for the EVM, which can also be converted into representative money based on the current pricing of the public Ethereum blockchain.

The recent movement of self-sovereign identity (SSI) [95] and verifiable credentials (VC) [84] in the context of DLTs also include approaches of privacy-preserving claim evaluations utilizing zero-knowledge proofs that can be adapted to be suitable for location-based information and services [308] [309].

7.6 Chapter summary

The evaluation chapter described the methodological approach with references to the relevant subsections. The chapter presented the seventeen (privacy) objectives and how each cryptographic method met them. An overview table of which cryptographic building blocks are used in the ten approaches categories, together with information on which primitive could enhance the schemes, is given in Appendix A.4.1. The necessary cryptographic building blocks and primitives are detailed in Section 4.4 and Appendix A for revision purposes. An exhaustive part of the chapter was dedicated to the computational evaluation of various approaches together with a comparison among them. The chapter closed with the additional use case of using the grid cells in a DLT environment.

Roads? Where we're going, we don't need... roads.

Back to the Future (1985)

8 Conclusion

As the comprehensive and open discussion on contact tracing and privacy during the pandemic in 2020 revealed, people care about data protection in general and their location privacy specifically. Also, it illustrated that the evaluation of specific data points to gain information and the protection of privacy are not mutually exclusive. This is in line with the results of this thesis.

The LBS use cases formed the basis for the 17 derived objectives. The objectives motivated the limitations of information sharing between the actors. The goal of not relying on a trusted third party pushed the requirements for the underlying schemes even further. The conversion of a point-in-polygon problem into a set membership calculation forms the basis for using privacy-preserving cryptographic schemes and primitives. This conversion addresses three goals:

Its usage for any arbitrarily shaped geofence and additional possibilities, such as distance or speed measuring. The set membership abstraction made it possible to use state-of-the-art cryptographic protocols. Before using S2 and Geohashes, possible encoding systems were evaluated. Computational efficiencies of the geo encodings were compared.

The thesis introduced the usage of inverted geofences to address particular objectives, such as revealing the actual positions of a target user when outside of a private area. It was shown that inverted geofences are beneficial for the majority of protocols.

The cryptographic protocols base their advantages on primitives such as hashing, asymmetric encryption, and commitment schemes. Additional to these primitives, higher-order protocols were used to address the objectives. Such protocols include homomorphic encryption, attribute-based encryption, zero-knowledge proofs, and cryptographic accumulators. The thesis showed that most of these tools and schemes are interconnected because of their similar underlying mathematical primitives. The thesis showed a use case for key-policy ABE to its usually used counterpart ciphertext-policy ABE. The schemes' usage was shown to be use case agnostic and ready for reactive and proactive LBS. The underlying set membership query also opens up the range of possible usages with protocols like Bluetooth, WiFi, and Cell-IDs. Some approaches are based on string

and pattern comparisons which can also be adopted.

Some approaches from the related work were extended to address specific objectives. Also, the privacy metric of LBS was expanded to address additional use cases better. As discussed in the motivation chapter, the thesis aimed mainly at using cases in which user identification is known to provide the service. These use cases include the carsharing and whereabouts scenarios.

8.1 Summary of results

The results of this thesis can be summed up as follows. The point-in-polygon query has introduced a construction based on grid cells and position parent values. Normalized and non-normalized forms of cell-based geofences enable set membership testing for a PIP query. This set membership construction can define any arbitrarily shaped geofence up to centimeter-level precision. Two geo encodings, i.e., S2 and Geohashes, have been compared. Both can be used for the presented use cases. S2 can form more accurate geofences, whereas Geohashes needless set members at the expense of less accuracy. The thesis has introduced inverted geofences, where its negative spaced geofence defines a geofence area, i.e., all cells around an area covering the entire globe except for the area of the wanted geofence. The inverted geofences help test target positions against areas with privacy in mind. The inverted geofences are usable with most cryptographic schemes and play out benefits when revealing a target position outside a pre-defined area in use cases where this is required.

Based on the set membership query for testing a position against an area, cryptographic primitives have been chosen to hide a client position from LBS services. The goals were to make TTP obsolete and enable services where the target user identification is known. The cryptographic schemes have been categorized into ten approaches mainly based on the dominant underlying tool. These tools were evaluated against 17 objectives derived from eight use cases. The thesis has introduced basic hashing for privacy-preserving geofencing based on grid cell parent testing. Current state-of-the-art protocols have been adapted for the geofencing context of the ten approaches. Related work, such as homomorphic encryption schemes, has been extended. Some use cases demanded privacy-preserving calculations, such as the area sizes and the numbers of geofences, without revealing their locations and reasons. Cryptographic tools were chosen to address these specific objectives. Proof of concept implementations have also been developed to evaluate memory sizes and calculation times computational demands as user interfaces for the policy-based approaches have been implemented and tested with users from the whereabouts use case. Some tools have emerged in the last years, and the privacy-preserving LBS techniques have been applied and tested with distributed ledger technologies and smart contracts. Additionally, some libraries used in the research community have been tested and compared for future applications. New cryptographic paradigms are still emerging.

8.1.1 Thesis statement

The *thesis statement* given in Section 2.7

> *One can design a geofence-based service with centimeter accuracy that respects*

> *the user's location privacy from a service provider without the need to introduce a trusted third party.*

has been addressed by the features of the ten cryptographic approaches, the geo encoding choices, and the evaluation of user positions against arbitrarily shaped virtual areas, i.e., geofences. As a result, the ten approaches (sometimes individually) match the 17 objectives. Enhancements to specific schemes can be made by combining features of encoding the data, for example, using cryptographic hashing to hide certain location information from an LBS service. The thesis showed that there is no need to introduce a trusted third party. Even the approach to blurring location data to make services more privacy-preserving is not necessary.

8.1.2 Addressed research questions

The following research questions were motivated by the use cases and introduced in Section 2.7.2. This is a summary of how the findings of this thesis address these questions.

How can a geofence with an arbitrary shape be encoded? The thesis motivates to represent geofences using normalized grid cells with an S2 or Geohash geo encoding. The encodings support accurate representations of any actual world shape or size, such as city borders, street networks, and cell tower coverage areas. The cell sizes and amounts can be adjusted according to the positioning accuracy. The thesis motivated this by introducing different use cases. Additional functionalities, such as privacy-preserving distance and speed measuring, were presented based on the mentioned encodings.

How can the position be checked against the encoded geofence? The position is represented by an S2 point or Geohash and its parent values. The set membership test determines if the position is inside the geofence or not. The geo encoding and the set membership tests were chosen to motivate its usage in privacy-preserving cryptographic schemes.

How can the encoding guarantee evaluation of exact positions? An S2 point has roughly the size of a square centimeter and has 30 parent values. The high-precision use is enabled by the normalized geofence representation, which can utilize the S2 point area on the edges of the respective geofence area.

How can location privacy be achieved even with knowing the user's identity? The thesis chooses and compares ten cryptographic schemes which support a set membership evaluation in a privacy-preserving manner. The cryptographic schemes were evaluated against a set of 17 objectives that were derived from the use cases. The use cases were partly chosen because the example services would know the identity of the target user. The approaches address the location privacy aspects. The schemes mainly support these aspects without needing a trusted third party. Sometimes a setup before the service consumption is required.

What methods are there to evaluate a masked position against a masked geofence? The thesis categorizes these privacy methods into ten meta-schemes: cryptographic hashing, probabilistic methods, homomorphic encryption, multi-party computation,

private set intersection, cryptographic accumulators, zero-knowledge proofs, attribute-based encryption, private pattern matching, and policy-based approaches.

How can the introduction of a trusted third party be avoided? Except for homomorphic encryption, all cryptographic schemes can be utilized to achieve this. In Section 5.3.6, a possible combination of an XORing bit mechanism along with a homomorphic multiplication would enable this scheme to bypass a trusted third party.

What additional features can the privacy-preserving mechanisms and geo-encodings address? The thesis shows applications for the privacy-preserving evaluation of limiting area sizes and grid cell numbers, together with additional features, such as privacy-preserving speed and distance measurement. This is particularly relevant for contact-tracing applications, as discussed in academia and the general public during the 2020 pandemic.

Can the presented approaches run on current hardware? The evaluation chapter shows that some approaches run on current hardware, including mobile devices and low-powered computers. Algorithmic performance advances and processor development should enable all schemes relevant for integration into future LBS services.

How could services and mobile OS providers use the methods? Section 5.5 describes how the presented approaches and schemes can be integrated into existing service infrastructures.

What are the differences between LBS classes? The approaches can be used in a variety of LBS classes as defined in Chapter 3, Tables 3.1 and 3.2. These classes include pro-active and re-active, self-, uni-, and bi-directional, indoor and outdoor, and network- and terminal-centric LBS. The set membership foundation enables the usage of the schemes also in environments in which WiFi-SSIDs or Bluetooth-IDs are used for positioning the target clients, such as indoor use cases.

How are these methods categorized in a general location-privacy taxonomy? The taxonomy referenced in Chapter 4 would categorize the presented methods under classical security or protocols. As current state-of-the-art schemes are used, including paradigms in cryptography, the term *classical* is inappropriate. Terms such as cryptographic protocols and schemes are more suitable and forward-looking, especially with quantum computing on the horizon.

What are attack vectors, and how can they be avoided? Inferred information collected through additional data points about targets or the environment may reveal information over time. However, this can be avoided by regularly updating setups and changing security and key values.

8.1.3 Contributions

The thesis' concepts and designs make scientific contributions to the following list (extract).

- Arbitrarily shaped geofence evaluation with high accuracy.
- Putting together privacy-preserving approaches based on cryptographic schemes.
- Privacy examples using inverted geofences.
- Introduction of grid cell hashing methods.
- Introduction of approach utilizing XOR, ZKP, and HE.
- Example for the usage of the rare Key-Policy ABE method.
- Notion of privacy gardens within policy-based schemes.
- Extension of geo encoding alphabets for private pattern matching.
- Making trusted third parties obsolete.
- Usable schemes in pro-active and re-active LBS.
- Privacy-preserving distance and speed measuring as additional features.
- Utilization of approaches for location-based access control.
- Comparison and evaluation of use-case-specific and agnostic methods.
- Evaluation of S2 cell parent approach.
- Evaluation of different geofence encodings.
- Evaluation of various cryptographic libraries, including ABE and ZKP.
- Evaluation of performance and memory consumption.
- Extension to the related work methods, such as NEXUS-Parallel.

As stated above, this subsection summarizes how the *thesis statement* is addressed, how the *research questions* were answered, and lists the most essential *contributions*.

8.1.4 Out of scope

As mentioned in Chapter 2, the thesis *does not* contribute to the following research fields, and these topics rely on advancements in the respective areas.

- Improving positioning accuracy
- Location spoofing resistance
- Advancements in user anonymity or pseudonymity
- Position blurring as a privacy method
- Measures for hiding user positions from intelligence agencies

In the context of positioning technologies, it has to be emphasized again that the ten approaches and the geo encodings described were also chosen to be adaptable to technologies such as Bluetooth and Wi-Fi.

8.2 Outlook and future research

After presenting the next steps to realize the thesis' core elements, an overview of future privacy-preserving use cases is given. Besides the use cases, open research questions in applied security schemes for privacy preservation and the combination with existing protocols are listed below.

8.2.1 Next steps

Next to the applied schemes, the thesis includes concepts based on cryptographic schemes that are not yet established in service providers' application development life cycle. Examples of such schemes are ZKPs, CA, and ABE. Robust and easy-to-use libraries must

be developed and tested for security vulnerabilities to use these approaches in a privacy-preserving scenario. At the same time, knowledge about these paradigms needs to be spread among developers to have a chance of appropriate use and integration into service workflows. Besides this technical information, the impact on their social and personal life demands a public discussion on the availability of privacy-preserving features. The introduction and the subsequent discussions on the differences and effects of various privacy approaches of contact tracing and exposure notification apps stand out as a recent example of such a public dialogue [60]. A university education in applied security schemes can be a starting point for introducing state-of-the-art and upcoming privacy schemes.

8.2.2 Further use cases

The thesis focuses mainly on the use cases presented in Chapter 2 and the derived objectives. Additional use cases to which privacy-preserving schemes are of use are described below, see also [19].

Current contact tracing and exposure notification apps that implement privacy-preserving protocols do not offer health authorities evaluation mechanisms for knowing which places people got infected the most. The thesis showed how to enable geofences for different location categories, such as public areas, campuses, and school grounds, shopping areas, offices, or industrial sites. With a privacy-preserving geofencing approach, these categories could be collected together with infection rates without revealing users' private information. Also, the approaches can add a layer of privacy to apps that crowdsource user opinions about locations, such as [310] or that use geolocation information for messaging [311]. The approaches also enable indoor use cases in which other positioning technologies are relevant [312] [313]. Parking lots and logistics evaluation or adaptable vehicle insurance are also use cases for integrating privacy. It is also possible to offer discounts for services when the target user is in particular areas, much like the carsharing use case. This can be summarized as location-sensitive billing. Additionally, a mobile operator could offer dynamic location credentials to users.

8.2.3 Future research

The following approaches and ideas were collected during the thesis process. They can be considered future research topics for applied privacy-preserving protocols and location-based services.

In Garzon et al. [77], a virtual reality interface was used to create 3D geofences. The schemes presented could generally be extended to a 3D model to include height information. With the advent of quantum computers, the specific usages of the ten cryptographic techniques have to be reevaluated and extended to withstand future attacks by introducing new countermeasures to be post-quantum secure. The author is aware of the fact that post-quantum secure protocols already exist.

The search for new protocols that support, for example, *indistinguishability obfuscation* (iO), which hides a program's implementation but allows users to run it, would enable a wide variety of applications. This protocol is considered one of the crown jewels of modern cryptography. Recent advances suggest that it is feasible [314]. With iO, it is

possible to construct other cryptographic primitives, including classical ones such as public-key cryptography and more exotic ones such as *deniable encryption* and *functional encryption*. Other protocols to be looked at and to be used in privacy schemes are for example, *ring signatures*, *blind signatures* (for an example scheme, see Appendix A.1.7), *verifiable random functions*, and *verifiable delay functions*. The area of *proof of location* and *proof of presence* protocols in security contexts might be another field of research worth investigating.

The current discussion about privacy-preserving techniques in the context of contact tracing apps has led to the belief or opinion that privacy stands in the way of computing relevant information about super-spreading events. Kolesnikov et al. showed in 2017 [315] that there are possibilities for PSI protocols for *multi-party sets*. That means that groups of individual clients can also be evaluated concerning location privacy aspects of people using privacy-preserving contact tracing apps. The approaches shown in this thesis can be adapted to assess multi-party sets of Bluetooth Low Energy IDs or other forms of simple set elements.

The grid cell approach could be used to derivate secret keys from each cell value. With an inverted geofence and its secret keys, the encrypted position of a target client can be revealed on the server-side each time one of the inverted cell keys decrypts the position in situations the target is outside of a private area.

Further research in the *indistinguishability of ciphertexts* along with symmetric encryption schemes, could be helpful in privacy protocols. In asymmetric encryption schemes, a cryptosystem that decouples the public key from the private key information would be beneficial in the eyes of the author for security and privacy use cases. The public key information is currently present in the private key part. This thesis states that such a cryptosystem could enhance privacy-preserving evaluation methods without trusted third parties.

8.3 A practical zero-knowledge proof

As introduced in Chapter 1, the *Where is Waldo?* analogy is comparable to the objectives of this thesis. Below, two approaches are shown on how to convince someone that you know the location of Waldo in such a *Wimmelbild* picture. These methods are described by Naor et al. [316] in 1998 and present an everyday example of demonstrating *having knowledge* in a zero-knowledge fashion to non-scientists or cryptographers.

The first one is relatively straightforward; copy the picture, cut out Waldo, and show it to the verifier. With this *proof*, the verifier can be convinced that you know where Waldo is and can use the original picture to find him herself.

The second approach is illustrated in Figure 8.1. Here you use a large piece of paper at least twice the dimensions of the Waldo picture, and you cut out a small hole in the middle. Afterward, you place the Where-is-Waldo picture underneath it and move it to Waldo's position. Again, with this *proof*, the verifier should be convinced that you know where Waldo is without disclosing the known location after pulling out the picture underneath the larger paper. Afterward, the verifier can also use the original image to

[1] Image source: The Waldo image of Chapter 1 is used.

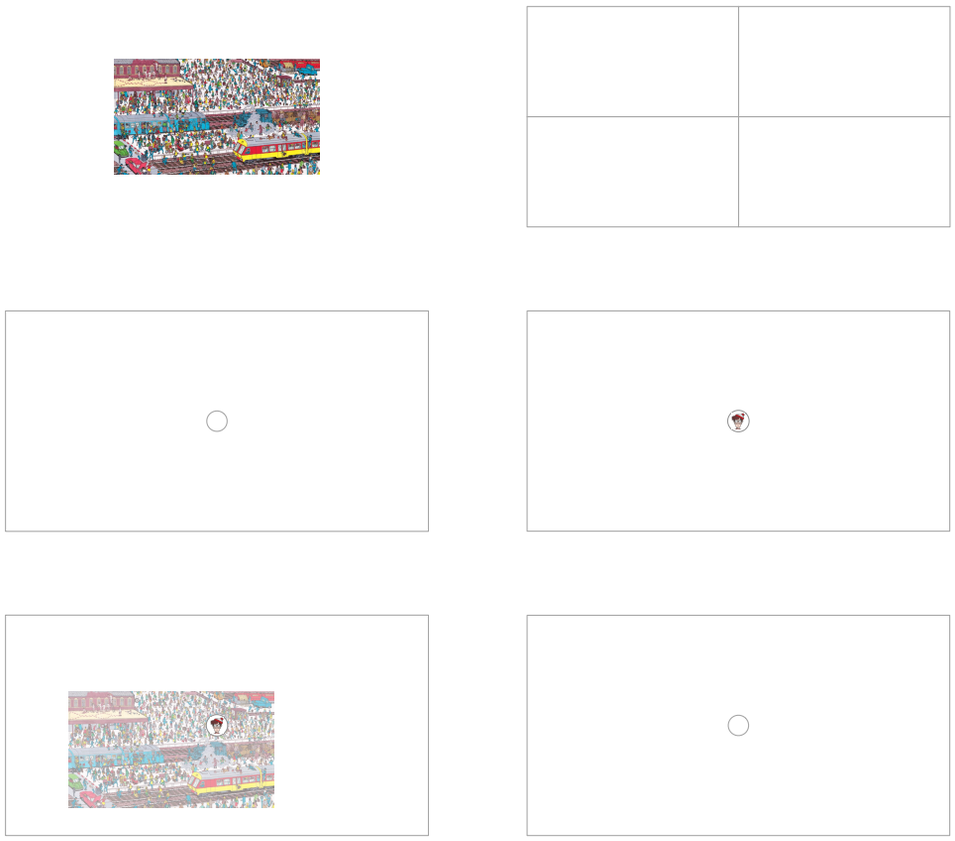


Figure 8.1: Where is Waldo? using a zero-knowledge proof.[1]

play the game.

Toto, I've got a feeling we're not in Kansas anymore.

The Wizard of Oz (1939)

A

Cryptographic Primitives and Schemes

The following cryptographic primitives and schemes are relevant for understanding the listed approaches and categories.

A.1 Mathematical foundations and cryptographic primitives

This section describes early and foundational cryptographic schemes and primitives as well as some of their underlying mathematical roots [90].

A.1.1 Vernam's one-time pad

A *Vernam's one-time pad* is defined as follows. Let $n \in \mathbb{N}$ and $M := C := \{0,1\}^n$. The randomized encryption $E()$ which encrypts a message $m \in M$ by XORing ($\oplus$) it bitwise with a chosen random and uniform bit sequence $k \leftarrow \{0,1\}^n$ of the length $|m| = n$.

$$E(k,m) := k \oplus m \tag{A.1}$$

$D()$ decrypts a ciphertext $c \in C$ by XORing it with the same k.

$$D(k,c) := k \oplus c \tag{A.2}$$

This is a perfect secure encryption algorithm because it contains maximum randomness in each new message. It is not apparent to an attacker whether the same message has been sent more than once since a new encryption key is randomly selected each time. It has the same length as the message itself [90]. However, the necessity to transfer this random encryption key for every new message makes this scheme impractical. An adversary could only gain knowledge of the length of the messages. This can be prevented by padding the messages with irrelevant data blocks, as applied in other

encryption schemes.

Based on Shannon's information theory [98], an encryption $E()$ is perfectly secret if the cipher text c from a set C and the message m from a set M are independent, i.e., the probability distribution of $M \cdot C$ is the product of each individual distribution M and C, see Equation A.3.

$$prob(m, c) = prob(m) \cdot prob(c), \text{ for all } m \in M, c \in C \tag{A.3}$$

A.1.2 Fields

A *field* is a set on which standard arithmetic operations are defined, i.e., addition, subtraction, multiplication, and division. $\mathbb{Z}$ is the set of integers. A *finite field* is a set with finitely many elements. The number of elements is also called the *order* of the field, mostly q. In cryptography, the order of the field is a prime number, i.e., the field has a prime order. A finite field of order q only exists if $q = p^k$ for a prime p and positive integer k. $\mathbb{F}_{17}$ is the field for the modular arithmetic example below. They are called *prime fields*.

A.1.3 Modular arithmetic

In 1801, Carl Friedrich Gauss developed the modern approach to modular arithmetic. An arithmetic system is based on integers, i.e., $\mathbb{Z}$. An everyday example is a 12-hour clock. Its set would be described as $\mathbb{Z}_{12}$. The following characteristics are essential for modern cryptography, such as public-key encryption, signatures, and zero-knowledge proofs.

- Multiplication and addition happen within a field.
- Divisions are done by modular inversion.
- The modular inverse works if the modulus is prime.
- A prime modulus is therefore used in cryptographic primitives.

Example for a modular inversion, in prime field $\mathbb{F}_{17} = \{0, 1, 2, 3, ..., 16\}$ [317].

$$2 \bmod 17$$

$$2^{-1} \bmod 17 = 1$$

$$2 \cdot x \bmod 17 = 1$$

$$x = 9$$

Examples for basic arithmetic operations (+,-,·,and / mod 17) in prime field $\mathbb{F}_{17}$ [317]

$$5 + 15 = 3$$

$$5 + 12 = 0$$

$$3 - 5 = 3 + (-5) = 3 + 12 = 15$$

$$4 \cdot 5 = 3$$

$$5 \cdot 7 = 1$$

$$3/5 = 3 \cdot (5^{-1}) = 3 \cdot 7 = 4$$

A *multiplicative group* is defined by multiplicating elements of $\mathbb{F}_p$. $(\mathbb{Z}/_p\mathbb{Z})^x$ is a multiplicative group of the prime field $\mathbb{F}_p$. A *generator g* produces a multiplicative group, i.e., reaching the elements of the field. 3 is a suitable generator for the finite field $\mathbb{F}_{17}$ as it produces 16 elements in the multiplicative group, $3^x \bmod 17, x = \{0, .., 15\}$.

A.1.4 Diffie Hellman

The following protocol is a practical solution for the key distribution problem based on public-key cryptography. The well-known protocol was published in 1976 by Diffie and Hellman [207] [90].

- Generate two publicly known values, a large prime modulus p and a generator g with $1 < g < p - 1$.
- Andy picks a random a, computes $c = g^a \bmod p$, and sends c.
- Becky chooses a random b, computes $d = g^b \bmod p$, and sends d to Andy.
- Andy computes the shared key $k = d^a \bmod p = (g^a)^b \bmod p = g^{ab} \bmod p$.
- Becky computes the same shared key with $k = c^b \bmod p = (g^b)^a \bmod p = g^{ba} \bmod p$.
- Both have the same shared secret key k, also known as a session key and can start their secure communication.

The example showcases why the Diffie-Hellman key agreement is also called *exponential key exchange*. The security assumption is that the modular approach is hard to brute force. Therefore, it is also called the *discrete logarithm assumption*, as

$$5^x = 125$$

is easy to compute, and

$$5^x \bmod 23 = 19$$

is hard to compute for a given field. *Blinded commitments* make it even more secure.

A.1.5 RSA cryptosystem

The key setup of the famous Rivest–Shamir–Adleman (RSA) public-key cryptosystem is briefly described below [90].

- Andy chooses two large prime numbers, p and q. He computes the coprime modulus $n = p \cdot q$.
- Andy chooses a prime integer e, that is $1 < e < r$, with $r = (p - 1)(q - 1)$.
- Andy publishes his public key (n, e) to Becky.
- Andy computes his secret key (n, d), with $d \equiv e^{-1} \bmod r$. d is the modular multiplicative inverse of e mod r.
- Becky can encrypt a message m with the public key values, $m^e \bmod n = c$.
- Andy can decrypt the ciphertext c using his secret key values, $c^d \bmod n = m$.

The numbers $n, e,$ and d are referred to as *modulus*, *encryption exponent*, and *decryption exponent*. The prime numbers are at least 1024 bits long. Longer primes, up to 4096, are less likely to be broken by factorization. Prime numbers with a bit length of only 300 can be broken by affordable hardware.

A.1.5.1 RSA digital signatures

The basic signature scheme [90] is as follows.

- p and q are very large prime numbers such that q divides $p-1$.
- G_q is the subgroup of order q in $\mathbb{Z}_p^*$.
- g is a generator of G_q.
- $h : \{0,1\}^* \to \mathbb{Z}_q$ is a collision-resistant hash function.
- The signers secret key is chosen randomly $x \in_R \mathbb{Z}_q$.
- The public key is (p,q,g,y), where $y = g^x$.

A.1.5.2 Strong RSA assumption

The security of some approaches in this thesis is based on the strong RSA assumption, briefly described, for example, in Li et al. [225] in the context of RSA-based universal accumulators, as described in Sections 4.4.6 and 5.3.5. The assumption assumes that it is infeasible to solve the following. Given an RSA modulus N and a random $x \leftarrow \mathbb{Z}_N^*$ it is not possible to find $e > 1$ and $y \in \mathbb{Z}_N^*$ such that $y^e = x \bmod N$. It states the *one-way* property of RSA protocols [90].

A.1.6 Schnorr's identification protocol

These are the steps of an interactive proof of knowledge using a discrete logarithm [90].

- Percy, the prover, wants to prove to Violet, the verifier, that he knows x, the discrete logarithm of $y = g^x$ to the base g.
- Percy picks a random value $r \in_R \mathbb{Z}_q$, computes $a = g^r$ and sends a to Violet.
- Violet picks a random value $c \in_R \mathbb{Z}_q$ and sends it to Percy.
- Percy computes $b = r - cx$ and returns b to Violet.
- Violet checks whether $a \equiv g^b y^c$. For more details on this step, see below.

A.1.7 Blind signatures

In blind signature schemes, the signer and the message author are two different entities. Also, the verifier can be the signer without recognizing its signature. In an example scenario, a customer spends a digital coin in an online shop. A bank signs the coin. The shop should be able to verify the bank's signature. If the shop brings the digital coin to the bank, the bank should not recognize that the signature was done for a specific customer's coin, i.e., the bank *blindly* signs the digital coins [90]. *Schnorr's blind signature scheme* by Claus Schnorr is used in protocols, such as zero-knowledge proof systems, see Sections 4.4.7 and 5.3.6 [318].

In Schnorr's blind signature scheme, the verifier Violet gets a blind signature for a message m from the prover Percy by executing the following protocol [90].

- Percy randomly chooses $\bar{r} \in_R \mathbb{Z}$, he computes $\bar{a} := g^{\bar{r}}$, and sends it to Violet.
- Violet chooses randomly $u, v, w \in_R \mathbb{Z}, u \neq 0$, she computes $a = \bar{a}^u g^v y^w, c := h(m||a)$ and $\bar{c} := (c-w)u^{-1}$, and sends $\bar{c}$ to Percy.
- Percy computes $\bar{b} := \bar{r} - \bar{c}x$ and sends it to Violet.
- Violet verifies $\bar{a} = g^{\bar{b}} y^{\bar{c}}$, computes $b := u\bar{b} + v$, and gets the signature $\sigma(m) := (c,b)$ of m. The condition for the signature (c,b) is $c = h(m||g^b y^c)$.

A.1.8 Commitment schemes

A commitment scheme is a cryptographic primitive that allows a prover to commit to a chosen value or statement while keeping it hidden from the verifier. The prover can reveal the committed value at a later point in time. The idea is that the party committing to a value cannot change it afterward. Therefore it is *binding*. The *hiding* property of the scheme means that the prover's value given to the verifier is not revealed at this stage. The *opening* part of the scheme would allow the verifier to know the value, i.e., the commitment. The prover chooses the time to do this. This also depends on the protocol or scheme. One-way hash functions were first used as commitments. In the context of zero-knowledge proofs, new schemes are defined.

This commitment scheme defines the algorithms *commit()* and *open()* and has two properties, *binding* and *hiding*.

- $commit(m, r) = c$ Given a message m and a random value r, $commit()$ computes as output a commitment c that hides the message m. It should be hard to compute a message m' and a value r' that satisfies $commit(m', r') = commit(m, r)$. It is hard to convert the function $commit()$ to find m or r. This property of a commitment scheme is called *binding*, i.e., it binds the message m to the commitment c. The property *hiding* ensures that for a given commitment c, it is hard to compute any information about m.
- $open(c, m, r) = b$ Given a commitment c, a message m and a randomness r, the algorithm returns true if and only if $c = commit(m, r)$.

If both parties follow the protocol, the receiver will recover the committed value. This property is called the *viability*.

A.1.8.1 Pedersen commitment

A perfectly binding and information-theoretically hiding commitment scheme is based on a public-key encryption scheme. The *Pedersen commitment* scheme [90] is based on discrete logarithms and enables a sender to commit to a message $m \in \{0, ..., q-1\}$. The protocol is as follows.

- $setup()$ The receiver randomly chooses large prime numbers p and q such that q divides $p-1$. After that, she randomly chooses the generator values g and h from a subgroup $\mathbb{G}_q$ of order q in $\mathbb{Z}*_p, g, h \neq [1]$ The receiver sends $p, q, g,$ and h to the sender of the message.
- $commit()$ The sender of the message verifies that p and q are primes, that q divides $p-1$, and that g and h are generators of order q. For the commitment to the message m, she chooses $r \in \{0, ..., q-1\}$ at random, sets $c := g^r h^m$, and sends c to the receiver.
- $reveal()$ The sender sends r and m to the receiver of the message. She verifies that $c = g^r h^m$.

In a commitment scheme based on logarithms, the hiding property is *unconditional*, and the binding property depends on the difficulty of computing discrete logarithms. In schemes depending on quadratic residues, the binding property is unconditional, and the hiding property relies on the infeasibility of calculating the quadratic residues property. There are no commitment schemes that are *both* unconditional hiding and binding. For additional security implications and features, see [90].

A.2 Homomorphic encryption schemes

Several homomorphic encryption schemes and their respective homomorphic properties are detailed in this subsection.

A.2.1 ElGamal

The *ElGamal public key* encryption scheme is homomorphic with respect to the *multiplication of plaintexts* and *ciphertexts*. Additionally, the scheme is also homomorphic with respect to the *addition of random values*. The notation of [90] is used in this section.

Let p be a large random prime, $g \in \mathbb{Z}^*$ a primative root, $x \in \mathbb{Z}^*_{p-1}$ be a random exponent, and $y = g^x$. (pk, sk) is the key pair with the *secret key* $sk = (p, g, x)$ and the *public key* $pk = (p, g, y)$. A message $m \in \mathbb{Z} * p$ is encrypted by $E_{pk}(m, \omega) = (g^\omega, y^\omega m)$, with the randomly chosen element $\omega \in \mathbb{Z}_{p-1}$.

For the homomorphism, this means

$$E_{pk}(m \cdot m', \omega + \omega') = E_{pk}(m, \omega) \cdot E_{pk}(m', \omega') \tag{A.4}$$

because

$$(g^\omega, y^\omega m) \cdot (g^{\omega'}, y^{\omega'} m') = (g^\omega g^{\omega'}, y^\omega m y^{\omega'} m') = (g^{\omega+\omega'}, y^{\omega+\omega'} m m'). \tag{A.5}$$

ElGamal encryption is probabilistic, i.e., a single plaintext can be encrypted to *many* possible ciphertexts, with the consequence that a general ElGamal encryption produces an expansion in size from plaintext to ciphertext. The *random factor* in creating the ciphertext can help tackle some attacks regarding the *unlinkability* of encrypted or hashed values.

ElGamal can also be applied in a slightly different way. The message m is encoded as g^m and then encrypted with $\tilde{E}_{pk} = (g^\omega, y^\omega g^m)$. Instead of m, g^m is used as g is also known to the encrypting entity as it is part of the public key. This variant has the following homomorphic encryption with respect to the *addition of plaintexts* and also with respect to the *multiplication of ciphertexts* as well as the *addition of a random element*. In detail

$$\tilde{E}_{pk}(m + m', \omega + \omega') = \tilde{E}_{pk}(m, \omega) \cdot \tilde{E}_{pk}(m', \omega'). \tag{A.6}$$

A.2.2 Paillier

The *Paillier* scheme's homomorphic property [239] relies on the residue class group $\mathbb{Z}^*_{n^2}$, with n being an RSA modulus. q is called a quadratic residue modulo n if there exists an integer x such that

$$x^2 \equiv q \bmod n. \tag{A.7}$$

Its public key is constructed as $n = pq$ and $g \in \mathbb{Z}^*_{n^2}$. g being an element of order n, e.g., $g = [1 + n]$. The secret key is $\lambda := \operatorname{lcm}(p - 1, q - 1)$, the *least common multiplier* of $p - 1$ and $q - 1$, with $\lambda = \lambda(n)$ called the *Carmichael function* of n. Then the encryption of a

plaintext m is $E_{pk}(m, \omega) = g^m \omega^n$, for some random $\omega \in \mathbb{Z}_n^*$.

The homomorphic addition of two plaintexts is defined as the product of two ciphertexts that will decrypt to the sum of their corresponding plaintexts, see the Equation below.

$$E_{pk}(m + m', \omega \cdot \omega') = E_{pk}(m, \omega) \cdot E_{pk}(m', \omega') \tag{A.8}$$

The privacy-preserving location framework NEXUS [176], see Section 4.4.3, also uses the pseudo-homomorphic multiplication, which means that an encrypted plaintext can be raised to a constant k. The result will decrypt to the product of the plaintext and the constant.

$$D_{sk}(E_{pk}(m, \omega)^k \bmod n^2) = k \cdot m \bmod n \tag{A.9}$$

In [176], it is shown that using the arithmetic addition and multiplication of the Paillier scheme can also be used to subtract values by adding a *negated* ciphertext homomorphically.

$$E_{pk}(m_1) \leq E_{pk}(m_2) = E_{pk}(m_1) \leq E_{pk}(m_2)^{-1} = E_{pk}(m_1 - m_2) \tag{A.10}$$

Re-encryption Homomorphic encryption schemes are inherent *malleable*. In the case of HE schemes, the property *malleable* is a desired feature. If it is possible to transform a ciphertext into another ciphertext and, when decrypted, relate to a plaintext, we speak of a *malleable encryption algorithm*. In the following example, the decrypting $f(m)$ may not reveal any information about m if $f()$ is not known.

$$\exists enc(m) \rightarrow enc(f(m)) \rightarrow dec(enc(f(m))) = f(m) \tag{A.11}$$

It has to be noted that in *both* the ElGamal and the Paillier encryption systems, it is possible to change the random factor ω *after* the message has been encrypted using E_{pk}. By that, the ciphertext c is being re-encrypted to c', either by adding new randomness $(\omega + \omega')$ in the ElGamal system or by multiplying new randomness $(\omega \cdot \omega')$ in the Paillier system. In both cases, the plaintext is not changed [90]. This behavior can be used to support additional privacy functionalities described in detail in Chapter 5.

A.2.3 Boneh–Goh–Nissim

The cryptosystem by Boneh, Goh, and Nissim (BGN) [224] is needed to understand the PSI-X protocol by Carpent et al. [179] in Section 5.3.4.3.

In BGN, the additive homomorphic property is the same as in the ElGamal scheme, but only one multiplication is allowed. That is why the scheme is called *somewhat homomorphic* [319].

The system relies on the hard-to-factor number n, which order is a composite number from an elliptic curve group. In ElGamal and other systems, the group order needed to be prime.

A curve for the scheme can be generated as follows.

- Pick two primes q and r.
- Publish $n = qr$.
- Find a small integer l, such that $4ln - 1$ is a prime p.
- Let E be an ellipctic curve $y^2 = x^3 + x$ over $\mathbb{F}_p$.
- Since $p \equiv 3 (\text{mod } 4)$ the curve is supersingular with $\#E(\mathbb{F}_p) = p + 1 = 4ln$, and thus there is a point $P \in E(\mathbb{F}_p)$ or order n.
- Such a point can be computed by choosing a random $P' \in E(\mathbb{F}_p)$ and setting $P = [4l]P'$.

The BGN scheme can be described as follows, based on Freeman [319].

- $gen()$ Choose large primes q, r and set $n = qr$. Take a supersingular elliptic curve $E/\mathbb{F}_p$ with a point P of order n and let $\mathbb{G} = m\langle P \rangle$. Choose $Q' \leftarrow \mathbb{G}\{\infty\}$ and set $Q = [r]Q'$; then Q has the order q. Let $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mu_n \subset \mathbb{F}_p^2$ be the modified Weil pairing construction [320]. Output the public key $pk = (E, \hat{e}, n, P, Q)$ and the secret key $sk = q$.
- $enc(pk, m)$ Choose $t \leftarrow [1, n]$ and output $C = [m]P + [t]Q$.
- $dec(sk, C)$ Compute $\tilde{P} = [q]P$ and $\tilde{C} = [q]C$, with output is $m' = log_{\tilde{P}}\tilde{C}$.
- $add(pk, C_1, C_2)$ $t' \leftarrow [1, n]$ with output $C' = C_1 + C_2 + [t']Q \in \mathbb{G}$.
- $mult(pk, C_1, C_2)$ $u \leftarrow [1, n]$ with output $D = \hat{e}(C_1, C_2) \cdot e(Q, Q)^u \in \mu_n$.
- $dec'(sk, D)$ Compute $\tilde{g} = g^q$ and $\tilde{D} = D^q$, with output $\log_{\tilde{g}}\tilde{D}$.
- $add'(pk, D_1, D_2)$ $u \leftarrow [1, n]$ with output $D' = D_1 \cdot D_2 \cdot h^u$.

A.3 Oblivious transfer

A building block for some MPC schemes is the oblivious transfer. In this section, the basic ones are described.

A.3.1 1-out-of-2 oblivious transfer

An *oblivious transfer (OT)* protocol enables the receiver to request *one* value (information) of *two* from the sender, but the sender is oblivious of which value was requested, i.e., sent. The protocol for 1-out-of-2-OT is as follows.

- Andy has two messages m_1 and m_2 and wants to send precisely one to Becky. Becky does not want Andy to know which one she receives.
- Andy generates an RSA key pair, with modulus N, the public exponent e, and the private exponent d.
- Andy also generates two random values x_1 and x_2 and sends them together with his public modulus N and his public exponent e to Becky.
- Becky picks b to be 0 or 1 and selects the first or second x_b.
- She generates a random value k and blinds x_b by calculating $v = (x_b + k^e) \text{ mod } N$. The result is sent to Andy.
- Andy does not know which x_b Becky chose. She calculates for both of her values two ks: $k_0 = (v - x_0)^d \text{ mod } N$ and $k_1 = (v - x_1)^d \text{ mod } N$. One of these will be equal to Becky's k, which only she will be able to decrypt. The other value will be random.
- Andy combines the two secret messages with each of the possible ks: $m'_0 = m_0 + k_0$ and $m'_1 = m_1 + k_1$ and sends both to Becky.
- Becky knows which of these two messages can be unblinded with k, which means she can compute one of the messages using $m_b = m'_b - k$.

A.3.2 1-out-of-n oblivious transfer

The generalization of the 1-out-of-2 OT protocol is a 1-out-of-n-OT, where Andy has n values out of x_1 to x_n, and Becky has an index of $b = \{1, .., n\}$. 1-out-of-∞-OT is used to construct a version of an *oblivious pseudorandom function* (OPRF) [177].

A.3.3 Oblivious pseudorandom function

In Section 4.4.1.3, the notion of a *pseudorandom generator* (PRG) was introduced. An additional cryptography primitive based on PRG is a *pseudorandom function* (PRF). The difference is that a PRF guarantees that all outputs seem to be random, no matter what the inputs are. Practical PRF takes a domain-specific input string and a hidden random value to produce an identical output string after several runs. The output can be considered random by using an arbitrary input string and a uniform distributed seed.

In an *oblivious pseudorandom function* (OPRF), the two-party protocol computes a joined PRF value $f()$ without seeing any secret input strings from the other party, i.e., Andy's input x or Becky's secret key k. The PRF value $f_k(x)$ is calculated *oblivious*. Its result is only known to the receiver, Becky. This primitive is a basic block in MPC for private set intersection protocols between untrusted parties.

Analog Kolesnikov et al. [321] define an OPRF semi-formally as follows. The protocol has two parties, a receiver and a sender. The receiver has input r. The sender outputs s, a random seed. The output of the pseudorandom function $PRF(s, r)$, as well as the input r, is only known to the receiver. The seed s is only known to the sender. OPRFs are used for PSI protocols in Section 5.3.4.

A.4 Fiat–Shamir heuristic

The technique proposed by Amos Fiat and Adi Shamir in 1986 [322] creates a digital signature based on the *interactive* proof of knowledge (IPoK) in Section 4.4.7, thus making it possible to convert it into a *non-interactive* ZKP. The initial IPoK must be based on a *publicly* available random value, which is done by a so-called trusted setup. The heuristic is secure under the assumption that a random oracle exists. It uses a hash function as a random oracle in step three of the following algorithm.

- Andy wants to prove that he knows x, the discrete logarithm of $y = g^x$ to the base g.
- He picks a random $v \in \mathbb{Z}_q^*$ and computes $t = g^v$ and sends t to Becky.
- Andy computes $c = h(g, y, t)$, with the cryptographic hash function $h()$.
- Andy computes $r = v - cx$. The proof is (t, r), where r is an exponent of g, calculated modulo $q - 1$, not modulo q.
- Any verifier knowing the public value can check whether $t \equiv g^r y^c$. Because the verifier knows g, y, and c and gets the proof values t and r. The verifier computes $g^r y^c$, which is $g^{v-cx} g^{x^c}$, which is $g^{v-cx} g^{cx}$, which is g^v, and that is t as computed in Step 2 and known as the value of the proof in Step 4.

A.4.1 Cryptographic primitives used in approaches

Table A.1 shows which cryptographic primitives form the basis for each of the ten approach categories. As can be seen in Chapter 5, the boundaries between the ten presented

categories are fluid. The tables show that specific cryptographic primitives and tools are used in different approaches, such as cryptographic hashing, RSA- and Diffie-Hellman-based public key systems, commitment schemes, or the homomorphic properties of ElGamal. The category terms used for the approaches are either based on the vocabulary of the related work, as seen in Chapter 4, or based on the most important primitive used, such as accumulators. Also, the boundary between primitives and higher concepts is not strict since the building blocks are based on the same cryptographic principles, such as the factorization of prime values or field arithmetic.

Table A.1: Overview of approaches, primitives, and algorithms

	1: Grid cell hashing	2: Probabilistic methods	3: Homomorphic encryption	4: Multi-party computations	5: Private set intersections	6: Cryptographic accumulators	7: Zero-knowledge proofs	8: Attribute-based encryption	9: Private pattern matching	10: Policy-based approaches
Vernams one time pad (OTP)	●	○	○	○	○	○	○	○	○	○
One way function	●	◐	○	◐	◐	◐	◐	◐	○	○
XOR	○	◐	◐	○	○	○	●	◐	○	○
Diffie Hellman	○	○	○	●	○	○	●	○	○	○
RSA	○	○	○	○	○	●	●	○	○	○
Cryptographic hashing	●	●	○	●	●	●	◐	◐	○	○
(H)MAC	●	◐	○	◐	◐	◐	◐	◐	○	○
Pseudorandom generators	●	○	○	◐	◐	○	○	○	○	○
Salt, pepper, nonce	●	●	●	●	●	◐	◐	◐	○	○
Key deriv./stretch./strength.	●	○	○	○	○	◐	◐	◐	○	○
Message digest	●	○	○	○	○	○	○	○	○	○
HOTP/TOTP	●	○	○	○	○	○	○	○	○	○
2-universal hash function	○	○	○	○	●	○	○	○	○	○
Bloom filter	○	●	○	○	○	○	○	○	○	○
Cuckoo hashing/filter	○	●	○	●	●	○	○	○	○	○
Merkle Trees	○	○	○	○	○	●	○	○	○	○
ElGamal	○	○	●	○	●	○	◐	●	●	○
Paillier	○	○	●	○	○	○	○	○	○	○
Boneh-Goh-Nissim	○	○	○	○	●	○	○	○	○	○
Oblivious transfer	○	○	○	○	●	○	○	○	○	○
Oblivious pseudor. function	●	○	○	●	●	○	○	○	○	○
Dot product protocol	○	○	●	○	○	○	○	○	○	○
zkSNARKS, Bulletproofs	○	○	○	○	○	○	●	○	○	○
Fiat Schamir heuristic	○	○	○	○	○	○	●	○	○	○
Pedersen commitment	○	○	○	○	○	●	○	○	○	○
KP-ABE, CP-ABE	○	○	○	○	○	○	○	●	○	○
(Geo)XACML	○	○	○	○	○	○	○	○	○	●
Inverse geofences	◐	◐	◐	◐	◐	◐	◐	◐	◐	◐

You can't handle the truth!

A Few Good Men (1992)

B

Publications

Some of the ideas, methods, and results that contribute to this thesis have been published in scientific publications, journals, and books and have been presented at national and international conferences and workshops. Research has been carried out in the research projects mentioned in the dedicated section. Therefore, the results of this thesis are partially contained in the following scientific publications.

Towards Location-based Services in a Cloud Computing Ecosystem [29]

Abstract: With the introduction of inexpensive mobile devices, which are capable of using location-based services (LBS), the popularity of such applications has grown rapidly. A lot of these applications are available to the public and used by them on a daily basis. In addition to this trend cloud computing was also rapidly adopted by professionals and private consumers alike. The project TRESOR1 evaluates and develops cloud computing solutions for the health sector domain. In this context location-based services are being researched. The paper presents different scenarios, which were all deducted from the current project work. The different areas such as location-based access control (LBAC), geofencing, mobile devices, geotagging, etc. are in the scope of the work in progress.

Innovative Architektur für sicheres Cloud Computing: Beispiel eines Cloud-Ecosystems im Gesundheitswesen [323]

Abstract: Der vorliegende Artikel stellt eine innovative Cloud Computing-Architektur vor, die eine Reihe von Herausforderungen im Einsatz von Cloud Computing adressiert: Datensicherheit, Konformität mit rechtlichen und organisatorischen Richtlinien sowie die Interoperabilität von Cloud-Lösungen. Diese ungelösten Herausforderungen bewirken, dass in vielen Branchen mit umfangreichen Anforderungen die Hauptvorteile von Cloud Computing, also Kostenreduzierung und höhere Flexibilität, nicht genutzt werden können. Besonders deutlich wird dies am Beispiel des Gesundheitswesens, wo es einen hohen Bedarf an sicheren und rechtskonformen Cloud Computing-Lösungen gibt. Zum Abschluss des Artikels wird die Evaluierung der vorgestellten Cloud Computing-Architektur im Rahmen des TRESOR Forschungsprojekts dargestellt. Im TRESOR Projekt werden ausgewählte medizinische Anwendungsfälle unter Nutzung dieser Architektur,

eingebettet in ein Cloud Computing-Ecosystem, realisiert.

Towards a Federated Cloud Ecosystem: Enabling Managed Cloud Service Consumption [324]

Abstract: While cloud computing has seen widespread usage, there exist domains where the diminishing of management capabilities associated with cloud computing prevent adoption. One such domain is the health sector, which is the focus of the TRESOR1 project. Enabling cloud computing usage under strict compliance constraints such as enterprise policies and legal regulations is the goal of TRESOR. The main approach consists of a distributed cloud proxy, acting as a trusted mediator between cloud consumers and service providers. In this paper we analyze issues which arise within the TRESOR context and show how an architecture for a proposed ecosystem bypasses these issues. The practicability of our solution is shown by a proof of concept proxy implementation. As all components of the architecture will be part of our proposed cloud ecosystem, we provide a holistic and generic proposal to regain management capabilities in cloud computing.

Deriving a Distributed Cloud Proxy Architecture for Managed Cloud Service Consumption [325]

Abstract: Businesses adopting Cloud Computing often have to comply with strict constraints, such as enterprise policies and legal regulations. From these compliance issues arise the need to enable managed cloud service consumption as a prerequisite for adoption. As we have shown before, the proposed TRusted Ecosystem for Standardized and Open cloud-based Resources (TRESOR) cloud ecosystem can achieve management of cloud service consumption. In this paper we motivate and derive the architecture of the distributed TRESOR cloud proxy from technical, business and legal requirements within the context of the TRESOR project. We apply a derivation method where we evaluate the impact of each incremental architecture decision separately. This process enables researchers with supplementary requirements to adapt the intermediate derivations within other contexts in flexible ways.

Enabling Location-based Services on Stationary Devices using Smartphone Capabilities [326]

Abstract: In recent years, location-based services became more and more popular. They serve information based on a user's or device's location. Many of these services are accessible over web interfaces via browsers, which would make them also interesting for stationary devices. However, these don't possess good positioning capabilities necessarily. There are still many of them that are connected to the Internet in a wired way, thus not being able to use WLAN positioning or similar approaches. Static devices usually rely on IP geolocation, which approximates their position only coarsely. Smartphones, on the other hand, come with inbuilt positioning technologies that allow to pinpoint the device and its user with very high accuracy. A potential solution to this dilemma is a connection between different device types. Highly accurate position information can subsequently be transferred in an automated and seamlessly integrated manner from a smartphone to a stationary device.

TRESOR - Towards the Realization of a Trusted Cloud Ecosystem [32]

Abstract: The TRESOR project enables cloud computing solutions for the German health sector. This sector deals with sensitive medical information and is in general not suitable for current cloud-based solutions, which are lacking appropriate privacy and security

features. The project evaluates and proposes new architectural components to address these shortcomings. These will be combined into a secure and trustworthy ecosystem that will enable the health industry and other sectors to take advantage of cloud computing. The architecture consists of components, such as a marketplace, a broker, a proxy and a PaaS-platform. TRESOR addresses privacy and data protection issues and aims at providing a standardized solution with reduced lock-in effects that can also be used in other domains. In this paper the specific tasks and the architecture of these components are presented, important challenges of the TRESOR project are highlighted and preliminary results, such as a secure transfer protocol, and policy integration are shown.

A Template-Based Policy Generation Interface for RESTful Web Services [33]

Abstract: Cloud computing solutions imply chances for economic ad- vantages concerning investment, administration and maintenance costs. On the downside these advantages are paid with a loss of autonomy; the service providers often predetermine configuration and authorization functionalities. The increase of participating actors represents recent privacy, security and legal issues for service providers and users. The different interests of all involved stakeholders raise a need for distributed access control functionalities, which consider the various restrictions of the stakeholders. The presented work designs and realizes a web interface, service users can use to express fine-grained access control policies concerning their resources. The increase of RESTful online services is addressed by a template approach that serves as a basis for the policy interface. A particular focus is set on the eXtensible Access Control Markup Language (XACML), a standard for distributed access control. Following the XACML standard the web interface is realized within the XACML component model. Users' requirements are retrieved via the web interface and translated into a complete XACML policy. The generated policies are tested for syntactic and semantic correctness as well as usability.

Enabling Location-based Policies in a Healthcare Cloud Computing Environment [31]

Abstract: In a multi-stakeholder cloud computing environment, data access control is of essential importance. Nowadays, it is usually handled in and deployed by every single cloud service on its own which makes the configuration of fine-grained access privileges cumbersome and economically expensive. In this paper, we introduce a novel cloud ecosystem architecture featuring an overall lightweight data access control model. This model is enabling data access policies based on location information of service consumer devices. We apply our architecture in the sensitive healthcare domain, which itself comprises multiple parties with complex data access privileges. Here, we define high-level requirements driven from current data protection regulations and guidelines as well as practice requirements in this area, which we address in the design of our architecture. We implement and test the main components. The results demonstrate the feasibility of our architecture and the applicability of our approach even in the healthcare application domain.

Securing Mobile Cloud Data with Personalized Attribute-based Meta Information [190]

Abstract: With the spread of fast mobile Internet connections, such as 3G and LTE and the increasing processor power of mobile devices accessing cloud computing services on- the-go is common among all users. Sharing private information with friends and

family members are options of popular cloud services, such as storage and social media services. But recent headlines show that the access to private information is often not sufficiently secured on the service level. The approach presented in this paper aims to use attribute-based meta-information to secure data on the level of files without relying on additional functionality of third-party services. A mobile device app is used to access and alter the meta-information. Attribute-based encryption mechanisms secure the private data and define access policies for friends and other users simultaneously.

Indoor Mapping for Location-based Policy Tooling using Bluetooth Low Energy Beacons [34]

Abstract: Most service providers and data owners desire to control the access to sensitive resources. The user may express restrictions, such as who can access the resources, at which point in time and from which location. However, the location requirement is difficult to achieve in an indoor environment. Determining user locations inside of buildings is based on a variety of solutions. Moreover, current access control solutions do not consider restricting access to sensitive data in indoor environments. This article presents a graphical web interface based on OpenStreetMap (OSM), called Indoor Mapping Web Interface (IMWI), which is designed to use indoor maps and floor plans of several real-world objects, such as hospitals, universities and other premises. By placing Bluetooth Low Energy (BLE) beacons inside buildings and by labeling them on digital indoor maps, the web interface back-end will provide the stored location data within an access control environment. Using the stored information will enable users to express indoor access control restrictions. Moreover, the IMWI enables and ensures the accurate determination of a user device location in indoor scenarios. By defining several scenarios the usability of the IMWI and the validity of the policies have been evaluated.

Proximity-based Services in Mobile Cloud Scenarios using Extended Communication Models [54]

Abstract: The progress in positioning technologies and the distribution of mobile devices with data communication capabilities promote the idea of providing proximity-based services. Proximity-based services deliver information and trigger actions, based on the location of users or devices. Recently, such applications became more popular in different facilities, such as shops, museums, and hospitals. Nevertheless, in most systems the service adaption is based solely on the location of a single user making the request. The presence or absence of other users is not considered. Furthermore, there is a need for a system, which can support the extension for different use cases without the need to change the program logic. The location of users is obtained by state-of-the-art wireless radio frequency technologies. How such a system can be designed for accessing mobile cloud data within a cloud computing ecosystem, as well as its feasibility is shown.

Dynamic Location Information in Attribute-based Encryption Schemes [197]

Abstract—Attribute-based encryption (ABE) allows users to encrypt (cloud) data with fine-grained Boolean access control policies. To be able to decrypt the ciphertext, users need to have a private key with the associated attributes. If the attributes satisfy the formula, the plaintext can be recovered. In this paper, ABE is extended with dynamic attributes. This allows attributes to be added to an existing private key. A server component named Attribute Authority is introduced. By using these dynamic attributes, it is now possible to have the decryption depend on data that changes often, such as location information

of a mobile device. Two schemes were developed that convert location data into usable ABE attributes. To demonstrate our results, an Android application was implemented and evaluated in a field test.

Semantic Location Information in Access Control Policy Models [30]

Abstract: The rise of mobile applications and their utilization of location-based services in recent years paved the way for new and useful applications, which are an integral part of our day-to-day life. Usually with the introduction of new applications and service domains the security implications are falling short in the beginning. The usage of location information as a security feature is discussed within the research community but practical applications are rare. Accessing data from a mobile device is common practice these days. The involvement of context information within access policies is on the rise. Here often time constraints are evaluated when accessing files. The usage of other context information also includes location aspects in access control schemes. Viable policies need to be constructed for security purposes. The construction of location policies is a task, which requires expert knowledge about the structure and details of the involved technologies. The aim of this publication is to simplify the creation of access control policies using contextual user information by using semantic representations of location attributes, e.g., the human understandable notion of, e.g., Sydney, California, or the campus of the University of Oxford, needs to be transferred into a location-based access control policy.

Applying Attribute-based Encryption on Publish Subscribe Messaging Patterns for the Internet of Things [189]

Abstract: With the advent of the Internet of Things (IoT), communication between connected machines has become necessity. We simulate the communication of IoT by short-lived instant messaging for group communication. Group communication security requires such measures as group forward and backward secrecy and perfect forward secrecy. We satisfy these security measures by using a group controller and Attribute-based Encryption (ABE) to encrypt data on update procedures. The communication overhead is outsourced to a mediating MQ Telemetry Transport broker. Thus, we decrease the costs for group joins and leaves to $\Theta(1)$. The number of attributes used in the system are reduced to $O(log(N))$, where N represents the maximum number of members. We provide an intuitive approach to fit the maximum number $N = 2k$ members to our requirements and to increase the maximum size of members, if needed by $N = 2k + 1$.

Applied Attribute-based Encryption Schemes [256]

Abstract: The advent of new cryptographic methods in recent years also includes schemes related to functional encryption. Within these schemes Attribute-based Encryption (ABE) became the most popular, including ciphertext-policy and key-policy ABE. ABE and related schemes are widely discussed within the mathematical community. Unfortunately, there are only a few implementations circulating within the computer science and the applied cryptography community. Hence, it is very difficult to include these new cryptographic methods in real-world applications. This article gives an overview of existing implementations and elaborates on their value in specific cloud computing and IoT application scenarios. This also includes a summary of the additions the authors made to current implementations such as the introduction of dynamic attributes.

privGardens - Semantic Privacy Areas in Location-based Data Protection Policies

[38]

Abstract: The paper describes the current situation and techniques used for carrying out unannounced doping tests. The current system ADAMS deals with sensitive information, such as the whereabouts and test results of top tier athletes around the word. The ADAMS system itself is not transparent about the process of access to private data. The presented concept of the PARADISE project focuses on the privacy and usability aspects of the location information. A dedicated device is described which can be queried by a doping control officer (DCO) to get the location of an athlete who should be tested. The notion of privacy gardens, called privGardens, is introduced in order to increase privacy aspects of the given use case.

Datenschutz und Dopingkontrollen [37]

Abstract: Können Privacy-Enhancing Technologies (PETs) tausenden Athleten zu mehr Privatsphäre verhelfen? Nicht erst seit den jüngsten Dopingenthüllungen stehen alle Athleten im Spitzensport unter einem Doping-Generalverdacht. Der Welt-Anti-Doping-Code hat die Unschuldsvermutung der Athleten schon seit Jahren abgeschafft. Sie können nur durch die Duldung massiver Eingriffe in ihre Privat- und Intimsphäre beweisen, dass sie »sauberen« Sport betreiben. Der Beitrag zeigt auf, dass diese massiven Eingriffe durch die Anwendung von PETs bald der Vergangenheit angehören können.

Identity and Access Management in a Doping Control Use Case [39]

Abstract: The PARADISE project is in need of a solution for a very specific set of requirements regarding identity and access management (IAM), in particular for the identity and attribute delegation and the authentication of users and roles within the competitive sport setting. Additionally, athletes have to share whereabouts information with doping control officers. Due to the complex relationship between entities in the doping control use case, innovative authorization solutions regarding location and role information are required. In this article, we present our response to this challenge using decentralized attribute-based access control (ABAC) and access control based on whereabouts using geofences.

Geofences on the Blockchain: Enabling Decentralized Location-based Services [121]

Abstract: A decentralized ride- or carsharing application is among the early proposals of what smart contracts on blockchains may enable in the future. To facilitate use cases in the field of location-based services (LBS), smart contracts need to receive trustworthy positioning information, and be able to process them. We propose an approach on how geofences can be defined in smart contracts, and how supplied positions can be evaluated on whether they are contained in the geofence or not. The approach relies on existing location encoding systems like Geohashes and S2 cells that can transform polygons into a grid of cells. These can be stored in a smart contract to represent a geofence. An oracle run by a mobile network provider can submit network-based positioning information to the contract, that compares it with the geofence. We evaluate the location encoding systems on their ability to model city geofences and mobile network cell position estimates and analyze the costs associated with storing and evaluating received oracle-positions in an Ethereum-based smart contract implementation. Our results show that S2 encodings perform better than Geohashes, that the one-time cost of geofence definition corresponds linearly with the number of grid cells used, and that the evaluation of oracle-submitted locations does not incur high costs.

PARADISE - Wie Ortungstechnologien den Datenschutz im Anti-Doping verbessern können [230]

Abstract: Mit den Ergebnissen des Forschungsprojekts PARADISE1wird eine neuartige, alternative Plattform zum derzeit eingesetzten System ADAMS definiert. In PARADISE wurde die Eignung moderner Positionierungstechnologien für den Einsatz im Anti-Doping-Anwendungsbereich evaluiert. Im Fokus stand hierbei die Frage, wie sich Datenschutz und Gebrauchstauglichkeit für AthletInnen mit unangekündigten Dopingkontrollen an verschiedenen Aufenthaltsorten vereinen lassen. Aktive AthletInnen wurden durch ihre Erfahrungen mit dem derzeitigen System motiviert, ein neue Lösung basierend auf dem derzeitigen Stand der Technik zu konzipieren. Der Artikel stellt diese Ergebnisse vor und zeigt, dass sich Datenschutz und Ortungstechnologien nicht ausschließen.

Full-text Search for Verifiable Credential Schemas on Distributed Ledgers [308]

Abstract: Self-sovereign Identity (SSI) powered by distributed ledger technologies enables more flexible and faster digital identification workflows, while at the same time limiting the control and influence of central authorities. However, a global identity solution must be able to handle myriad credential types from millions of issuing organizations. As metadata about types of digital credentials is readable by everyone on the public permissioned ledger with Hyperledger Indy, anyone could find relevant and trusted credential types for their use cases by looking at the records on the blockchain. To this date, no efficient full-text search mechanism exists that would allow users to search for credential types in a simple and efficient fashion tightly integrated into their applications. In this work, we propose a full-text search framework based on the publicly available metadata on the Hyperledger Indy ledger for retrieving matching credential types. The proposed solution is able to find credential types based on textual input from the user by using a full-text search engine and maintaining a local copy of the ledger. Thus, we do not need to rely on information about credentials coming from a very large candidate pool of third parties we would need to trust, such as the website of a company displaying its own identifier and a list of issued credentials. We have also proven the feasibility of the concept by implementing and evaluating a prototype of the full-text credential metadata search service.

Interactive Design of Geofences for Proactive Location-based Services in Smart Cities [77]

Abstract: Recently, location-based services for Smart Cities became capable to proactively notify mobile users about hyperlocal information or location-based reminders once users enter clearly-specified geographical regions, better known as geofences. Up to now, geofences are mostly designed by humans through the interactive design of circular or polygonal shapes on 2-dimensional digital maps. In future Smart City applications, geofences will not only represent flat regions but also arbitrary 3-dimensional spaces such as no-fly zones for drones. This work investigates the applicability of using head-mounted displays together with touch controllers to interactively design 3-dimensional geofences within a virtual 3D representation of the world. It presents and discusses a proof of concept that enables humans to examine, create, modify, move, scale, and rotate 3-dimensional geofences and to edit its meta information within a generated 3D virtual environment of the inner city of Berlin.

Distributed-Ledger-based Authentication with Decentralized Identifiers and Verifi-

able Credentials [309]

Abstract: Authentication with username and password is becoming an inconvenient process for the user. End users typically have little control over their personal privacy, and data breaches effecting millions of users have already happened several times. We have implemented a proof of concept decentralized OpenID Connect Provider by marrying it with Self-Sovereign Identity, which gives users the freedom to choose from a very large pool of identity providers instead of just a select few corporations, thus enabling the democratization of the highly centralized digital identity landscape. Furthermore, we propose a verifiable credential powered decentralized Public Key Infrastructure using distributed ledger technologies, which creates a straightforward and verifiable way for retrieving digital certificates.

Carpe diem. Seize the day, boys. Make your lives extraordinary.

Dead Poets Society (1989)

C

Supervised Theses

Some of the ideas, methods, and results that contributed to this thesis have been developed during the following supervised bachelor, master, and diploma theses. Parts of this thesis and code were therefore addressed and developed in the following monographs.

Graf, Thomas. Enabling Location-based Applications on Stationary Devices using Smartphone Capabilities. Master Thesis, Technische Universität Berlin, 2012

Raschke, Philip. A Template-based Policy Generation Interface for RESTful Web Services. Bachelor Thesis, Technische Universität Berlin, 2014

Denisow, Iwailo. Dynamic Geolocation Attributes within Functional Encryption Schemes. Bachelor Thesis, Technische Universität Berlin, 2015

Becker, Alex. Web-based semantically enriched crowd-sourced cloud computing information. Bachelor Thesis, Technische Universität Berlin, 2015

Förster, Alexander. Attribute-based Encryption in Publish Subscribe Messaging Patterns. Diploma Thesis, Technische Universität Berlin, 2015

El-Sobhy, Jasir. Proximity-based Services using Wireless Radio Frequency Technologies. Master Thesis, Technische Universität Berlin, 2015

Sharhan, Senan. Indoor Mapping using BLE-Beacons for Policy Tooling. Bachelor Thesis, Technische Universität Berlin, 2015

Ochunyi, Philip Barnabas. Location-based Cloud Storage using Geographic Policies. Bachelor Thesis, Technische Universität Berlin, 2016

Wierzchowski, Ron. An Access History Schema for Personal Cloud Resources. Bachelor Thesis, Technische Universität Berlin, 2016

Pöllabauer, Thomas Jürgen. Implementation of a Virtual Reality Interface for Geofencing. Bachelor Thesis, Technische Universität Berlin, 2017

Sahin, Anil. Smartphone-Based Public Transport Vehicle Identification Using GTFS. Master Thesis, Technische Universität Berlin, 2017

Fadhel, Mehdi Ben. Reducing Location Tampering in Indoor Positioning Systems. Master Thesis, Technische Universität Berlin, 2017

Samper, Martin. Privacy-Enhanced Reactive Indoor Positioning. Bachelor Thesis, Technische Universität Berlin, 2017

Taffo, Franck William. Konzipierung eines datenschutzfreundlichen Navigationssystems mittels isochronen Karten. Master Thesis, Technische Universität Berlin, 2018

Bayraktar, Ersin. Implementation and Evaluation of Privacy-Aware Geofence Verifications. Master Thesis, Technische Universität Berlin, 2019

Alharash, Ghiath. Design and Implementation of User Interface for Geocoding Systems. Master Thesis, Technische Universität Berlin, 2019

Gomez, Jessica. Increasing Trust and Transparency in Doping Control Processes using Distributed Ledger Technologies. Master Thesis, Technische Universität Berlin, 2019

That's all folks!

Porky Pig

D Projects

Some of the ideas, methods, and results that contributed to this thesis have been developed during the following research projects funded by national and international institutions, see brackets. To a certain extent, parts of this thesis are therefore contained in the reports.

D.1 Research projects

The author of this thesis researched his concepts and ideas in the context of the following research projects while being employed at the Technische Universität Berlin.

D.1.1 TRESOR

TRESOR is a project selected within the Trusted Cloud technology competition, which is going to be funded by the Federal Ministry of Economics and Technology (BMWi).

The main objective of TRESOR is the development of a cloud ecosystem that consists of an open platform for cloud services and a cloud broker. The first component provides a marketplace for cloud services based on well-established standards maximizing interoperability between different cloud providers. In that way, cloud services can be easily combined and integrated into several applications leading to cost savings and decreasing dependencies, so-called lock-in effects, to certain cloud providers. The cloud broker, on the other hand, mediates, combines, and allows access to those cloud services by taking enterprise guidelines, regulations by law, and security policies into consideration. Furthermore, the broker applies new concepts of location-aware service usage and data storage.

- Time Frame: 03/2012 - 02/2015
- Project partners:
  - T-Systems International GmbH
  - UBIRY GmbH
  - Deutsches Herzzentrum Berlin

  - Paulinenhaus Krankenanstalt e.V. Berlin
  - medisite Systemhaus GmbH
  - Technische Universität Berlin
- Funding by: Bundesministerium für Wirtschaft und Technologie
- Website: www.cloud-tresor.de

D.1.2 Curcuma

Curcuma was chosen as one of the IT projects in 2014 to be funded within the Software Campus initiative.

Software Campus integrates leading research and practical experience in a new type of concept in order to create a new generation of top managers and entrepreneurs with excellent IT backgrounds. (© Software Campus)

Curcuma (Cloud Computing Location Metadata) is a research project that aims at using location information and metadata structures within cloud computing environments to protect, enrich and structure files and limit access to information and cloud resources using dedicated containers.

The location information of mobile clients and server ecosystems comes into operation when accessing files. The location information on a mobile client is dynamic. This dynamic location attribute is considered when granting access to cloud data. The cloud data is secured with attribute-based encryption (ABE) policies which include viable location information within the access rule. The notion of dynamic attributes enriches this field of research. But not only will the location of the accessing device come into play within the Curcuma scenarios, but also the server location is considered when deciding where to store the data with regard to privacy and data protection concerns.

The cloud data itself will be enriched with semantic meta-information to bind the access information to personal user information and the content data itself. Access histories of highly sensitive data, such as personal health records (PHR), will be stored within the metadata structures of the container files. This should be seen as an optional feature which will be applied only to highly personal information.

A subproject of Curcuma is the Open Cloud Computing Map (OCCMap) which provides the basis for the (visual and non-visual) server location information of different cloud computing providers and interlinks to sources regarding data protection laws and other linked-data information. The data provided by the OCCMap will be offered in a structured semantic manner to be included in other open and linked data initiatives.

Contributions of Curcuma will be made in the fields of meta-data structures, semantically linked data, dynamic ABE security/access features, such as location-based access control (LBAC), as well as mobile cloud and mobile data scenarios.

- Time Frame: 03/2014 - 02/2016
- Project partners:
  - Telekom Innovation Laboratories
  - Technische Universität Berlin
- Funding by: Software Campus, Federal Ministry of Education and Research (BMBF)
- Website: www.curcuma-project.net

D.1.3 PARADISE

The privacy project PARADISE is funded by the Federal Ministry of Education and Research (BMBF). The project goal is to increase the confidentiality and privacy of athletes' whereabouts data in the context of national anti-doping controls. Together with our partners and the National Anti-Doping Agency (NADA) as an associated partner, we are improving the privacy aspects of national athletes by introducing a privacy-enhancing anti-doping environment.

- Time Frame: 01/2016 - 12/2017
- Project partners:
  - Fraunhofer FIT
  - Fraunhofer AISEC
  - gekko
  - UNiSCON
  - NADA (associated)
  - Technische Universität Berlin
- Funding by: Federal Ministry of Education and Research (BMBF)
- Website: www.privacy-paradise.de

D.1.4 Identity Chain

With IdentityChain, service providers like financial institutions will be able to provide their services to customers in a faster and cheaper way by verifying a customer's digital identity - as unforgeable proven by an attester - using a distributed identity management platform instead of running an own costly KYC process. Customers will have exclusive rights to grant or revoke service providers' access to particular attributes of their proven digital identity via an easy-to-use and well-designed mobile application.

- Time frame: 01/2018 - 12/2018
- Project partners:
  - Technische Universität Berlin
  - Telekom / T-Labs
  - ING Netherlands
  - CWI Netherlands
- Funding by: European Institute of Technology - Digital

D.1.5 Decentralized Identity Management System

DIMS envisions a decentralized Pan-European platform for identity information in the digital age. It enables an identity solution with the power and trust of distributed ledger technology, the principles of self-sovereignty, and in compliance with the relevant EU legislation. European service providers, as well as European citizens, will benefit from the integrated solution of the self-sovereign identity management system. On the one hand, to cut costs on how to use customer data to provide public services that will work across borders. On the other hand, for citizens to have a one-stop solution for their digital identity information. DIMS will support a regulated exchange of identity information among European service providers.

- Time Frame: 01/2019 - 12/2019

- Project partners:
  - Technische Unievrsität Berlin
  - OTP Bank
  - E-Group
  - Poste Italiane
  - CWI Netherlands
- Funding by: European Institute of Technology - Digital
- Website: www.dims-project.eu

D.1.6 Self-sovereign Identity für Deutschland

With five consortial partners and 15 additional associate partners, the ambitious SSI4DE project is bringing Self Sovereign Identity to Germany with one goal in mind: Empowering individuals to own and control their own digital identities. Funded by BMWi (Bundesministerium für Wirtschaft und Energie), SSI4DE is creating the modern infrastructure for the digital issuance of credentials from institutions and verifications of those credentials for service providers while giving individuals the power to share which credentials they want to share with the service providers.

SSI4DE covers a vast amount of industries and sectors. Banking, eGovernment, and student services are only a few of the many areas in which the large cross-disciplinary, multi-corporational, and multi-institutional SSI4DE team is working on. Privacy by design, powered by distributed ledger technologies, and keeping interoperability in mind, SSI4DE is technologically equipping Germany for the digital era.

- Time Frame: 06/2020 – 11/2020
- Project partners:
  - Main Incubator GmbH
  - Bundesdruckerei GmbH
  - esatus AG
  - Robert Bosch GmbH
  - Technische Universität Berlin
- Funding by: BMWi (Bundesministerium für Wirtschaft und Energie)
- Website: www.idunion.org

D.2 Supervised Student Projects

Some of the ideas, methods, and results that contributed to this thesis have been developed during the following supervised university courses. To a certain extent, parts of this thesis are therefore contained in the following results and written reports.

- Summer semester 2013: GeoXACML (with the participation of Iwailo Denisow)
- Summer semester 2014: TU-B Here (with the participation of Philip Raschke)
- Winter semester 2014/2015: TU-B Here 2
- Winter semester 2015/2016: Indoor Navigation
- Summer semester 2016: Privacy Geofencing (in the context of the PARADISE project)
- Winter semester 2016/2017: Train Prediction (in the context of the PARADISE project)
- Summer semester 2017: Semantic Geofencing (in the context of the PARADISE / eden(internally) projects)

- Winter semester 2017/2018: Blockchain-Identity (in the context of the IDChain project)
- Summer semester 2018: Self-sovereign Identity on Distributed Ledger Technology (in the context of the IDChain and DIMS projects)
- Winter semester 2018/2019: Usable Privacy Seminar (together with the Quality and Usability chair)
- Summer semester 2019: Payment using Self-sovereign Identities

Lists

List of Tables

List of Figures

Listings

Bibliography

[1] R. Golijan, "Arthur c. Clarke predicted GPS and satellite TV in 1956 (news on gzmodo.com)," 27-Jul-2010. [Online]. Available: https://gizmodo.com/arthur-c-clarke-predicted-gps-and-satellite-tv-in-1956-5597169.

[2] "WGS 1984 spatial reference website." [Online]. Available: https://spatialreference.org/ref/epsg/wgs-84/.

[3] L. Becker, "iOS 13: Abfrage von WLAN-Infos nur noch mit Standortfreigabe." [Online]. Available: https://www.heise.de/mac-and-i/meldung/iOS-13-Abfrage-von-WLAN-Infos-nur-noch-mit-Standortfreigabe-4505438.html.

[4] I. Pakalski, "Apple schwächt Werbe- und Tracking-Verbot für Kinder-Apps ab," 16-Sep-2019. [Online]. Available: https://www.golem.de/news/app-store-regeln-apple-schwaecht-werbe-und-tracking-verbot-fuer-kinder-apps-ab-1909-143872.html.

[5] L. Goode, "Mobile security firm says it defeated Strava's privacy feature with simple geometry," 07-Feb-2018. [Online]. Available: https://www.theverge.com/2018/2/7/16983408/strava-privacy-zones-wandera-security-geofence-safety-heat-maps.

[6] K. Opsahl, "Uber Should Restore User Control to Location Privacy," 07-Dec-2016. [Online]. Available: https://www.eff.org/deeplinks/2016/12/uber-should-restore-user-control-location-privacy.

[7] A. J. Hawkins, "New startup aims to bring controversial scooter tracking system to more cities," 23-Sep-2019. [Online]. Available: https://www.theverge.com/2019/9/23/20872421/lacuna-startup-mds-scooter-tracking-ladot-data-collection.

[8] A. Merz, "Sicherheitslücke ermöglicht Erstellung von Bewegungsprofilen," 07-Oct-2016. [Online]. Available: https://www.golem.de/news/lovoo-sicherheitsluecke-ermoeglicht-erstellung-von-bewegungsprofilen-1610-123642.html.

[9] J. Reardon, "What the Huq?" 25-Oct-2021. [Online]. Available: https://blog.appcensus.io/2021/10/25/what-the-huq/.

[10] J. Cox, "Location Data Firm Got GPS Data From Apps Even When People Opted Out," 25-Oct-2021. [Online]. Available: https://www.vice.com/en/article/5dgmqz/huq-location-data-opt-out-no-consent.

[11] S. Zuboff, *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. Profile Books, 2019.

[12] K. Conger, "Facebook says it's not making friend suggestions based on your location after all," 29-Jun-2016. [Online]. Available: https://techcrunch.com/2016/06/28/facebook-says-its-not-making-friend-suggestions-based-on-your-location-after-all/.

[13] J. Cox, "Memorials of the dead ask people not to play Pokémon Go on their grounds," 12-Jul-2016. [Online]. Available: http://www.theverge.com/2016/7/12/12166492/pokemon-go-sensitive-locations-arlington-emetery-holocaust-museum.

[14] “Geo-fence on Definitions.net.” [Online]. Available: https://www.definitions.net/definition/geo-fence.

[15] Meg, “Strava Privacy Zones,” 18-Aug-2021. [Online]. Available: https://support.strava.com/hc/en-us/articles/115000173384-Privacy-Zones.

[16] N. Mueller, “Strava’s privacy zone feature is almost worthless,” 07-Mar-2018. [Online]. Available: https://medium.com/@nickolaus/stravas-privacy-zone-feature-is-almost-worthless-f47c5cebfece.

[17] P. B. the Elder, “Netherlandish Proverbs,” 1559. [Online]. Available: https://artsandculture.google.com/asset/WwG8mD89xbELbQ.

[18] M. Handford, “Where is Waldo?” [Online]. Available: https://whereswaldoemotionally.files.wordpress.com/2013/09/waldo11.jpg.

[19] S. Frattasi and F. Della Rosa, *Mobile positioning and tracking: From conventional to cooperative techniques*. John Wiley & Sons, 2017.

[20] A. F. Westin, *Privacy and freedom*. Ig Publishing, 1967.

[21] B. Rainey, E. Wicks, and C. Ovey, *Jacobs, White and Ovey: the European convention on human rights*. Oxford University Press, USA, 2014.

[22] “European Court of Human Rights.” [Online]. Available: https://www.echr.coe.int/Pages/home.aspx?p=basictexts.

[23] “General Data Protection Regulation (GDPR),” 25-May-2018. [Online]. Available: https://gdpr.eu/tag/gdpr/. [Accessed: 29-Sep-2021].

[24] W. Stallings, *Information Privacy Engineering and Privacy by Design: Understanding Privacy Threats, Technology, and Regulations Based on Standards and Best Practices*. Pearson Education, 2019.

[25] A. Cavoukian and others, “Privacy by design: The 7 foundational principles,” *Information and privacy commissioner of Ontario, Canada*, vol. 5, 2009.

[26] B. Friedman, P. Kahn, and A. Borning, “Value sensitive design: Theory and methods,” *University of Washington technical report*, no. 2–12, 2002.

[27] B. Friedman, P. H. Kahn, A. Borning, and A. Huldtgren, “Value sensitive design and information systems,” in *Early engagement and new technologies: Opening up the laboratory*, Springer, 2013, pp. 55–95.

[28] “IPEN - Internet Privacy Engineering Network.” [Online]. Available: https://edps.europa.eu/data-protection/ipen-internet-privacy-engineering-network_en.

[29] S. Zickau and A. Küpper, “Towards Location-based Services in a Cloud Computing Ecosystem,” in *Ortsbezogene Anwendungen und Dienste - 9. Fachgespräch der GI/ITG-Fachgruppe Kommunikation und Verteilte Systeme*, 2012, vol. 9, pp. 187–190.

[30] S. Zickau, “Semantic Location Information in Access Control Policy Models,” in *Proceedings of the Location-based Applications and Services (LBAS 2015)*, 2015.

[31] S. Zickau, D. Thatmann, T. Ermakova, J. Repschläger, R. Zarnekow, and A. Küpper, “Enabling Location-based Policies in a Healthcare Cloud Computing Environment,” in *Proceedings of the 3rd IEEE International Conference on Cloud Networking (IEEE CloudNet)*, 2014, pp. 333–338.

[32] S. Zickau, M. Slawik, D. Thatmann, S. Uhlig, I. Denisow, and A. Küpper, “TRESOR - Towards the Realization of a Trusted Cloud Ecosystem,” in *Trusted Cloud Computing*, 2014, pp. 141–157.

[33] P. Raschke and S. Zickau, “A Template-Based Policy Generation Interface for RESTful Web Services,” in *On the Move to Meaningful Internet Systems: OTM 2014 Workshops*, 2014, vol. 8842, pp. 137–153.

[34] S. M. Sharhan and S. Zickau, “Indoor mapping for location-based policy tooling using Bluetooth Low Energy beacons,” in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2015, pp. 28–36.

[35] A. Matheus and J. Herrmann, “Geospatial extensible access control markup language (GeoXACML),” *Open Geospatial Consortium Inc. OGC*, 2008.

[36] “eXtensible Access Control Markup Language (XACML) Version 3.0,” 22. January 2013. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html.

[37] T. Herber, M. Jentsch, and S. Zickau, “Datenschutz und Dopingkontrollen,” vol. 41, pp. 427–433, Jul. 2017.

[38] S. Zickau, “privGardens - Semantic Privacy Areas in Location-based Data Protection Policies,” in *13. GI/KuVS-Fachgespräch Ortsbezogene Anwendungen und Dienste*, 2016.

[39] M. Schanzenbach and S. Zickau, “Identity and access management in a doping control use case,” *Datenschutz und Datensicherheit-DuD*, vol. 41, no. 12, pp. 724–728, Dec. 2017.

[40] “The Anti-Doping Administration & Management System (ADAMS),” 29-Sep-2021. [Online]. Available: https://www.wada-ama.org/en/what-we-do/adams.

[41] “PARADISE - Privacy-enhancing And Reliable Anti-Doping Integrated Service Environment.” [Online]. Available: https://privacy-paradise.de/.

[42] “WADA Confirms Attack by Russian Cyber Espionage Group,” 13-Sep-2016. [Online]. Available: https://www.wada-ama.org/en/media/news/2016-09/wada-confirms-attack-by-russian-cyber-espionage-group.

[43] “Glympse website.” [Online]. Available: http://glympse.com/.

[44] S. P. Fraiberger and A. Sundararajan, “Peer-to-peer rental markets in the sharing economy,” *NYU Stern School of Business Research Paper*, 2017.

[45] “Share Now website,” 06-Nov-2021. [Online]. Available: https://www.share-now.com/.

[46] “Miles Mobility website,” 06-Nov-2021. [Online]. Available: https://miles-mobility.com/.

[47] “Zipcar website.” [Online]. Available: https://www.zipcar.com/.

[48] “Turo website,” 06-Nov-2021. [Online]. Available: https://turo.com/gb/en.

[49] “Getaround website,” 06-Nov-2021. [Online]. Available: https://getaround.com/.

[50] “Snappcar website,” 06-Nov-2021. [Online]. Available: https://www.snappcar.de/.

[51] E. D. P. Supervisor, “Preliminary Opinion on privacy by design.” [Online]. Available: https://edps.europa.eu/sites/edp/files/publication/18-05-31_preliminary_opinion_on_privacy_by_design_en_0.pdf.

[52] C. Lorenz, “BMW: Neue digitale Funktionen für PHEV,” 25-Jun-2019. [Online]. Available: https://www.heise.de/autos/artikel/BMW-Ab-2020-neue-digitale-Funktionen-fuer-PHEV-4454339.html.

[53] M. Antonini, M. Ruggieri, R. Prasad, U. Guida, and G. F. Corini, “Vehicular remote tolling services using EGNOS,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 20, no. 10, pp. 3–8, 2005.

[54] J. El-Sobhy, S. Zickau, and A. Küpper, “Proximity-based Services in Mobile Cloud Scenarios using Extended Communication Models,” in *Proceedings of the 4th IEEE international conference on cloud networking (IEEE CloudNet)*, 2015.

[55] P. Ruppel, G. Treu, A. Küpper, and C. Linnhoff-Popien, “Anonymous User Tracking for Location-based Community Services,” in *Proceedings of the 2nd intl. Workshop on location and context-awareness*, 2006, pp. 116–133.

[56] A. Küpper and G. Treu, “From Location to Position Management: User Tracking for Location-based Services,” in *Tagungsband der ITG/GI-fachtagung ”kommunikation in verteilten systemen” (KiVS 05)*, 2005, vol. 2005.

[57] G. Treu, A. Küpper, and P. Ruppel, “Anonymization in Proactive Location Based Community Services,” in *Advances in Pervasive Computing. Adjunct Proceedings of the Third Intl. Conference on Pervasive Computing*, 2005.

[58] A. Küpper and G. Treu, “Efficient Proximity and Separation Detection among Mobile Targets for Supporting Location-Based Community Services,” *Mobile Computing and Communication Review (MC2R)*, vol. 10, no. 3, pp. 1–12, Jul. 2006.

[59] B. Johnson, “The Covid Tracing Tracker: What’s happening in coronavirus apps around the world,” 16-Dec-2020. [Online]. Available: https://www.technologyreview.com/2020/12/16/1014878/covid-tracing-tracker/.

[60] S. Landau, *People Count: Contact-Tracing Apps and Public Health*. MIT Press, 2021.

[61] N. Szabo, “The idea of smart contracts,” *Nick Szabo’s Papers and Concise Tutorials*, vol. 6, 1997.

[62] V. Buterin, “A next-generation smart contract and decentralized application platform,” *Ethereum Project White Paper*, 2014.

[63] H. Wymeersch, G. Seco-Granados, G. Destino, D. Dardari, and F. Tufvesson, “5G mmWave positioning for vehicular networks,” *IEEE Wireless Communications*, vol. 24, no. 6, pp. 80–86, 2017.

[64] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design science in information systems research,” *MIS quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[65] A. R. Hevner, “A three cycle view of design science research,” *Scandinavian journal of information systems*, vol. 19, no. 2, p. 4, 2007.

[66] P. J. Denning, “A new social contract for research,” *Communications of the ACM*, vol. 40, no. 2, pp. 132–134, 1997.

[67] P. Bellavista, A. Küpper, and S. Helal, “Location-Based Services: Back to the Future,” *IEEE Pervasive Computing*, vol. 7, no. 2, pp. 85–89, 2008.

[68] M. Labrador, K. Michael, and A. Küpper, “Special Issue on Advanced Location-based Services,” *Computer Communications*, vol. 31, no. 6, Apr. 2008.

[69] A. Küpper, G. Treu, and C. Linnhoff-Popien, “TraX: A Device-centric Middleware for Location-based Services,” *IEEE Communications Magazine, Special Issue on ”Advances in Service Platform Technologies for Next Generation Mobile Systems”*, vol. 44, no. 9, pp. 114–120, Sep. 2006.

[70] “Google Gofencing Developer Information.” [Online]. Available: https://developers.google.com/location-context/geofencing.

[71] “Lexico Geofence Definition.” [Online]. Available: https://www.lexico.com/en/definition/geofence.

[72] U. Bareth, A. Küpper, and B. Freese, “Geofencing and Background Tracking - The Next Features in LBS,” in *Proceedings of the 41th annual conference of the gesellschaft für informatik e.v. (INFORMATIK 2011)*, 2011, vol. 192.

[73] A. Küpper, F. Fuchs, M. Schiffers, and T. Buchholz, “Supporting Proactive Location-Aware Services in Cellular Networks,” in *Proceedings of the 8th IFIP WG 6.8 Conference on Personal Wireless Communications (PWC 2003)*, 2003.

[74] S. Rodriguez Garzon, B. Deva, G. Pilz, and S. Medack, “Infrastructure-assisted Geofencing: Proactive Location-based Services with Thin Mobile Clients and Smart Servers,” in *Proceedings of the 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, 2015, pp. 61–70.

[75] S. Rodriguez Garzon, D. Arbuzin, and A. Küpper, “Geofence Index: A Performance Estimator for the Reliability of Proactive Location-based Services,” in *18th IEEE International Conference on Mobile Data Management (MDM)*, 2017, pp. 1–10.

[76] S. Rodriguez Garzon, M. Elbehery, B. Deva, and A. Küpper, “Reliable Geofencing: Assisted Configuration of Proactive Location-based Services,” in *Proceedings of the 2016 IEEE International Conference on Mobile Services*, 2016, pp. 204–207.

[77] S. Rodriguez Garzon, T. Pöllabauer, S. Zickau, and A. Küpper, “Interactive Design of Geofences for Proactive Location-based Services in Smart Cities,” in *Proceedings of the 16th IEEE international conference on ubiquitous intelligence and computing (UIC 2019)*, 2019.

[78] S. Rodriguez Garzon and B. Deva, “Geofencing 2.0: taking location-based notifications to the next level,” in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014, pp. 921–932.

[79] A. Küpper, *Location-based Services: Fundamentals and Applications*. 2005.

[80] M. Herrmann, M. Hildebrandt, L. Tielemans, and C. Diaz, “Privacy in location-based services: An interdisciplinary approach,” *SCRIPTed*, vol. 13, p. 144, 2016.

[81] Y. Georgiadou, R. A. de By, and O. Kounadi, “Location Privacy in the Wake of the GDPR,” *ISPRS international journal of geo-information*, vol. 8, no. 3, p. 157, 2019.

[82] M. Hazas, J. Krumm, and T. Strang, “Location- and context-awareness,” *Proceedings of LoCA 2005*, 2006.

[83] A. Pfitzmann and M. Hansen, “A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management,” 2010.

[84] M. Sporny, D. Longley, and D. Chadwick, “Verifiable Credentials Data Model 1.0,” 05-Feb-2021. [Online]. Available: https://w3c.github.io/vc-data-model/.

[85] M. Backes, B. Pfitzmann, and M. Waidner, “A general composition theorem for secure reactive systems,” in *Theory of Cryptography Conference*, 2004, pp. 336–354.

[86] G. Danezis, J. Domingo-Ferrer, M. Hansen, J.-H. Hoepman, D. L. Metayer, R. Tirtea, and S. Schiffner, “Privacy and data protection by design-from policy to engineering,” *arXiv preprint arXiv:1501.03726*, 2015.

[87] Y. Lindell and B. Pinkas, “An efficient protocol for secure two-party computation in the presence of malicious adversaries,” in *Annual international conference on the theory and applications of cryptographic techniques*, 2007, pp. 52–78.

[88] Y. Lindell and B. Pinkas, “Secure two-party computation via cut-and-choose oblivious transfer,” *Journal of cryptology*, vol. 25, no. 4, pp. 680–722, 2012.

[89] S. Goldwasser and S. Micali, “Probabilistic encryption,” *Journal of computer and system sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[90] H. Delfs and H. Knebl, *Introduction to cryptography Principles and Applications*, 3rd ed. Springer, 2015.

[91] E. Barker, A. Roginsky, R. Blank, and P. D. Gallagher, “NIST Special Publication 800-133 Recommendation for Cryptographic Key Generation,” 2012.

[92] D. R. Kuhn, V. Hu, W. T. Polk, and S. H. Chang, “Introduction to Public Key Technology and the Federal PKI Infrastructure.” National Institute of Standards & Technology, 2001.

[93] R. Ross, M. McEvilley, and J. Oren, “Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems [including updates as of 1-03-2018].” Special Publication (NIST SP), National Institute of Standards; Technology, Gaithersburg, MD, 2018.

[94] M. Hansen, M. Jensen, and M. Rost, “Protection goals for privacy engineering,” in *2015 IEEE Security and Privacy Workshops*, 2015, pp. 159–166.

[95] A. Preukschat and D. Reed, *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*. Simon; Schuster, 2021.

[96] A. Kerckhoffs, *La cryptographie militaire, ou, Des chiffres usités en temps de guerre: avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef*. Librairie militaire de L. Baudoin, 1883.

[97] A. Kerckhoffs, “The information hiding homepage - Kerckhoffs’ principles from «La cryptographie militaire».” [Online]. Available: https://www.petitcolas.net/kerckhoffs/index.html.

[98] C. E. Shannon, “Communication theory of secrecy systems,” *Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

[99] International Telecommunication Union, Telecommunication Standardization Sectorm, “Security Architecture for Open Systems Interconnection (OSI) for CCITT Applications,” 1991.

[100] D. Bleichenbacher, “Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS# 1,” in *Annual International Cryptology Conference*, 1998, pp. 1–12.

[101] "Malleability (cryptography)," 31-Jan-2022. [Online]. Available: https://en.wikipedia.org/wiki?curid=87027.

[102] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes," in *Annual International Cryptology Conference*, 1998, pp. 26–45.

[103] O. Goldreich, *Foundations of cryptography: Volume 2, basic applications*. Cambridge university press, 2009.

[104] A. Toler, "How to Use and Interpret Data from Strava's Activity Map," 29-Jan-2018. [Online]. Available: https://www.bellingcat.com/resources/how-tos/2018/01/29/strava-interpretation-guide/.

[105] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 75–86.

[106] J. Reid, J. M. G. Nieto, T. Tang, and B. Senadji, "Detecting relay attacks with timing-based protocols," in *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, 2007, pp. 204–213.

[107] X. Bultel, S. Gambs, D. Gérault, P. Lafourcade, C. Onete, and J.-M. Robert, "A prover-anonymous and terrorist-fraud resistant distance-bounding protocol," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2016, pp. 121–133.

[108] E. Pagnin, G. Hancke, and A. Mitrokotsa, "Using distance-bounding protocols to securely verify the proximity of two-hop neighbours," *IEEE Communications Letters*, vol. 19, no. 7, pp. 1173–1176, 2015.

[109] U. Dürholz, M. Fischlin, M. Kasper, and C. Onete, "A formal approach to distance-bounding RFID protocols," in *International Conference on Information Security*, 2011, pp. 47–62.

[110] I. Boureanu, A. Mitrokotsa, and S. Vaudenay, "Practical and provably secure distance-bounding," *Journal of Computer Security*, vol. 23, no. 2, pp. 229–257, 2015.

[111] A. Hassidim, Y. Matias, M. Yung, and A. Ziv, "Ephemeral Identifiers: Mitigating Tracking & Spoofing Threats to BLE Beacons," 2016.

[112] A. Mitrokotsa, C. Onete, and S. Vaudenay, "Location leakage in distance bounding: Why location privacy does not work," *Computers & Security*, vol. 45, pp. 199–209, 2014.

[113] W. C. Salix alba, "The types of polygons," 2007. [Online]. Available: https://commons.wikimedia.org/wiki/File:Polygon_types.svg.

[114] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, Third Edition. Springer, 2010.

[115] "Wikipedia polygons," 31-Jan-2022. [Online]. Available: https://en.wikipedia.org/wiki/Polygon.

[116] W. C. Melchoir, "Crossing number algorith example," 2007. [Online]. Available: https://commons.wikimedia.org/wiki/File:RecursiveEvenPolygon.svg.

[117] J. Erickson, "The Jordan polygon theorem," *Computational Topology*, vol. 20, 2012.

[118] K. Hormann and A. Agathos, "The point in polygon problem for arbitrary polygons," *Computational geometry*, vol. 20, no. 3, pp. 131–144, 2001.

[119] W. C. Jim Belk, "Winding number examples." [Online]. Available: https://en.wikipedia.org/wiki/Winding_number#/media/File:Winding_Number_-2.svg.

[120] G. Niemeyer, "Geohash website," 2008. [Online]. Available: http://geohash.org.

[121] F. Victor and S. Zickau, "Geofences on the Blockchain: Enabling Decentralized Location-based Services," in *BlockSEA 2018, The 1st Workshop on Blockchain and Sharing Economy Applications, co-located with The 18th IEEE International Conference on Data Mining (ICDM 2018)*, 2018.

[122] "Mapcode website." [Online]. Available: https://www.mapcode.com/.

[123] "Open post code website." [Online]. Available: http://www.openpostcode.org/.

[124] "Natural area code website." [Online]. Available: http://www.nacgeo.com/nacsite/.

[125] "Universal Transverse Mercator coordinate system PROJ coordinate transformation software library." [Online]. Available: https://proj.org/operations/projections/utm.html?highlight=utm.

[126] "Open location code google github website." [Online]. Available: https://github.com/google/open-location-code.

[127] "HEALPix JPL/NASA website." [Online]. Available: http://healpix.jpl.nasa.gov.

[128] "What words." [Online]. Available: https://what3words.com/.

[129] s2geometry-io@googlegroups.com, Ed., "The S2 geometry website," 24-Nov-2021. [Online]. Available: https://s2geometry.io/.

[130] Wikimedia, "Longitude lines are perpendicular to and latitude lines are parallel to the Equator." 2005. [Online]. Available: https://en.wikipedia.org/wiki/Geographic_coordinate_system#/media/File:FedStats_Lat_long.svg.

[131] S. Xu and K. Kamath, "Dynamic geohash-based geofencing." Google Patents, 2016.

[132] S. J. Faber, "Variants of Privacy Preserving Set Intersection and their Practical Applications," PhD thesis, UC Irvine, 2016.

[133] "Geohash-int," 2014. [Online]. Available: https://github.com/yinqiwen/geohash-int.

[134] "Geohash-hilbert," 2016. [Online]. Available: https://github.com/tammoippen/geohash-hilbert.

[135] S. Heuberger, "Geohash-java." [Online]. Available: https://github.com/kungfoo/geohash-java.

[136] "S2 Geometry Library on Github." [Online]. Available: https://github.com/google/s2geometry.

[137] C. S. Perone, "Google's S2, geometry on the sphere, cells and Hilbert curve," 14-Aug-2015. [Online]. Available: https://blog.christianperone.com/2015/08/googles-s2-geometry-on-the-sphere-cells-and-hilbert-curve/.

[138] O. Procopiuc, “Geometry on the Sphere: Google’s S2 Library.” [Online]. Available: https://docs.google.com/presentation/d/1Hl4KapfAENAOf4gv-pSngKwvS_jwNVHRPZTTDzXXn6Q/view#slide=id.i0.

[139] “Sidewalk Labs Region Coverer.” [Online]. Available: https://s2.sidewalklabs.com/regioncoverer/.

[140] M. S. Andersen and M. B. Kjærgaard, “Towards a new classification of location privacy methods in pervasive computing,” in *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, 2011, pp. 150–161.

[141] B. Gedik and L. Liu, “A customizable k-anonymity model for protecting location privacy,” Georgia Institute of Technology, 2004.

[142] L. Liu, “From Data Privacy to Location Privacy: Models and Algorithms.” in *VLDB*, 2007, vol. 7, pp. 1429–1430.

[143] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, “L-diversity: Privacy beyond k-anonymity,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, pp. 3–es, 2007.

[144] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, “Geo-indistinguishability: Differential privacy for location-based systems,” in *Proceedings of the 2013 ACM SIGSAC conference on computer & communications security*, 2013, pp. 901–914.

[145] J. M. Bohli, D. Dobre, G. O. Karame, and W. Li, “PrivLoc: Preventing location tracking in geofencing services,” in *International Conference on Trust and Trustworthy Computing*, 2014, pp. 143–160.

[146] “Privacypatterns.org location granularity,” 11-Jan-2022. [Online]. Available: https://www.privacypatterns.org/patterns/Location-granularity.

[147] Z. Zhi and Y. K. Choong, “Anonymizing geographic ad hoc routing for preserving location privacy,” in *25th IEEE International Conference on Distributed Computing Systems Workshops*, 2005, pp. 646–651.

[148] S. Rodriguez Garzon, B. Deva, B. Hanotte, and A. Küpper, “CATLES: A Crowdsensing-supported Interactive World-scale Environment Simulator for Context-aware Systems,” in *Proceedings of the 2016 IEEE/ACM international conference on mobile software engineering and systems*, 2016, pp. 77–87.

[149] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, “The new casper: Query processing for location services without compromising privacy,” in *Proceedings of the 32nd international conference on very large data bases*, 2006, pp. 763–774.

[150] C. Roth, L. Hartmann, and D. Kesdoğan, “PARTS–Privacy-Aware Routing with Transportation Subgraphs,” in *Nordic Conference on Secure IT Systems*, 2017, pp. 86–101.

[151] M. L. Damiani, E. Bertino, and C. Silvestri, “Protecting location privacy against spatial inferences: The PROBE approach,” in *Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS*, 2009, pp. 32–41.

[152] E. Yigitoglu, M. L. Damiani, O. Abul, and C. Silvestri, “Privacy-preserving sharing of sensitive semantic locations under road-network constraints,” in *2012 IEEE 13th International Conference on Mobile Data Management*, 2012, pp. 186–195.

[153] M. L. Damiani, “Third party geolocation services in LBS: privacy requirements and research issues,” 2011.

[154] L. F. Cranor, "P3P: Making privacy policies more useful," *IEEE Security & Privacy*, vol. 1, no. 6, pp. 50–55, 2003.

[155] R. Barnes, M. Lepinski, A. Cooper, J. Morris, H. Tschofenig, and H. Schulzrinne, "An architecture for location and location privacy in internet applications," *Request for Comments*, vol. 6280, 2009.

[156] B. Deva, S. R. Garzon, and S. Schünemann, "A context-sensitive privacy-aware framework for proactive location-based services," in *2015 9th international conference on next generation mobile applications, services and technologies*, 2015, pp. 138–143.

[157] [Online]. Available: https://github.com/DP-3T.

[158] [Online]. Available: https://github.com/pepp-pt/pepp-pt-documentation/.

[159] [Online]. Available: https://github.com/DP-3T/dp3t-sdk-ios#introduction.

[160] [Online]. Available: https://github.com/DP-3T/dp3t-sdk-android#introduction.

[161] M. Herrmann, "Privacy in location-based services," 2016.

[162] M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," 2015.

[163] S. Kelly and S. Frankel, "Using hmac-sha-256, hmac-sha-384, and hmac-sha-512 with ipsec," RFC 4868, May, 2007.

[164] R. Pass and A. Shelat, "A Course in Cryptograph," *Theoretical Foundation of Cryptography*, pp. 68–71, 2010.

[165] O. GOLDREICH, S. GOLDWASSER, and S. MICALI, "How to construct random functions," *Journal of the Association for Computing Machinery*, vol. 33, no. 4, pp. 792–807, 1986.

[166] O. Goldreich, *Foundations of cryptography, volume 1, basic tools*. Cambridge university press, 2007.

[167] H. Krawczyk, M. Bellare, and R. Canetti, "RFC2104: HMAC: Keyed-hashing for message authentication." RFC Editor, 1997.

[168] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[169] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE/ACM transactions on networking*, vol. 8, no. 3, pp. 281–293, 2000.

[170] R. P. F. F. Rodler, "Cuckoo hashing," in *9th Annual European Symposium on Algorithms, ESA*, 2001.

[171] "Cuckoo hashing (wikipedia)," 04-Feb-2022. [Online]. Available: https://en.wikipedia.org/wiki/Cuckoo_hashing.

[172] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than bloom," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, 2014, pp. 75–88.

[173] C. Bösch, "An Efficient Privacy-Preserving Outsourced Geofencing Service Using Bloom Filter," in *2018 IEEE Vehicular Networking Conference (VNC)*, 2018, pp. 1–8.

[174] P. Ruppel and A. Küpper, "Geocookie: A space-efficient representation of geographic location sets," *Journal of Information Processing (JIP)*, vol. 22, no. 3, pp. 418–424, Jul. 2014.

[175] P. Palmieri, L. Calderoni, and D. Maio, "Spatial bloom filters: Enabling privacy in location-aware applications," in *International Conference on Information Security and Cryptology*, 2014, pp. 16–36.

[176] M. Guldner, T. Spieldenner, and R. Schubotz, "NEXUS: Using Geo-fencing Services without revealing your Location," in *2018 Global Internet of Things Summit (GIoTS)*, 2018, pp. 1–6.

[177] D. Evans, V. Kolesnikov, M. Rosulek, and others, "A Pragmatic Introduction to Secure Multi-Party Computation," *Foundations and Trends in Privacy and Security*, vol. 2, no. 2–3, pp. 70–246, 2018.

[178] E. De Cristofaro, P. Gasti, and G. Tsudik, "Fast and private computation of cardinality of set intersection and union," in *International Conference on Cryptology and Network Security*, 2012, pp. 218–231.

[179] X. Carpent, S. Faber, T. Sander, and G. Tsudik, "Private set projections & variants," in *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*, 2017, pp. 87–98.

[180] J. Benaloh and M. De Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Workshop on the Theory and Application of of Cryptographic Techniques*, 1993, pp. 274–285.

[181] N. Fazio and A. Nicolosi, "Cryptographic accumulators: Definitions, constructions and applications," *Paper written for course at New York University*, 2002.

[182] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 1997, pp. 480–494.

[183] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Annual International Cryptology Conference*, 2002, pp. 61–76.

[184] D. Derler, C. Hanser, and D. Slamanig, "Revisiting cryptographic accumulators, additional properties and relations to other primitives," in *Cryptographers' Track at the RSA Cconference*, 2015, pp. 127–144.

[185] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Annual International Cryptology Conference*, 1997, pp. 410–424.

[186] E. Morais, T. Koens, C. Van Wijk, and A. Koren, "A survey on zero knowledge range proofs and applications," *SN Applied Sciences*, vol. 1, no. 8, pp. 1–17, 2019.

[187] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: A new vision for public-key cryptography," *Communications of the ACM*, vol. 55, no. 11, pp. 56–64, 2012.

[188] L. Touati and Y. Challal, "Efficient CP-ABE Attribute/Key Management for IoT Applications," in *Computer and Information Technology (CIT), 2015 IEEE International Conference on*, 2015, pp. 343–350.

[189] D. Thatmann, S. Zickau, A. Förster, and A. Küpper, “Applying Attribute-based Encryption on Publish Subscribe Messaging Patterns for the Internet of Things,” in *Proceedings of the 8th IEEE International Conference on Internet of Things (iThings 2015)*, 2015, pp. 556–563.

[190] S. Zickau, F. Beierle, and I. Denisow, “Securing Mobile Cloud Data with Personalized Attribute-based Meta Information,” in *Proceedings of the 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, 2015, pp. 205–210.

[191] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data,” in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006, pp. 89–98.

[192] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *Security and Privacy, 2007. SP’07. IEEE Symposium on*, 2007, pp. 321–334.

[193] S. Jahid and N. Borisov, “PIRATTE: Proxy-based Immediate Revocation of ATTribute-based Encryption,” *arXiv preprint arXiv:1208.4877*, 2012.

[194] A. Sahai, H. Seyalioglu, and B. Waters, “Dynamic credentials and ciphertext delegation for attribute-based encryption,” in *Advances in Cryptology–CRYPTO 2012*, Springer, 2012, pp. 199–217.

[195] S. G. Weber, “Securing first response coordination with dynamic attribute-based encryption,” in *Privacy, Security, Trust and the Management of e-Business, 2009. CONGRESS’09. World Congress on*, 2009, pp. 58–69.

[196] N. Doshi and D. Jinwala, “Updating Attribute in CP-ABE: A New Approach,” *IJCA Proceedings on International Conference in Distributed Computing and Internet Technology 2013*, vol. ICDCIT, pp. 23–28, 2013.

[197] I. Denisow, S. Zickau, F. Beierle, and A. Küpper, “Dynamic Location Information in Attribute-based Encryption Schemes,” in *Proceedings of the 9th International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST 2015)*, 2015, pp. 240–247.

[198] S. G. Weber, “A Hybrid Attribute-Based Encryption Technique Supporting Expressive Policies and Dynamic Attributes,” *Information Security Journal: A Global Perspective*, vol. 21, no. 6, pp. 297–305, 2012.

[199] L. Scott and D. E. Denning, “A Location Based Encryption Technique and Some of Its Applications,” 2003.

[200] J. Daemen and V. Rijmen, *The design of rijndael: AES-the advanced encryption standard*. Springer, 2002.

[201] P. Grassi, M. Garcia, and J. Fenton, “Digital Identity Guidelines (Revised Draft),” National Institute of Standards; Technology, 2017.

[202] M. Dworkin, “Recommendation for Block Cipher Modes of Operation. Methods and Techniques,” *NIST Special Publication 800-38A2001 Edition*, 2001.

[203] P. FIPS, “113 Computer Data Authentication,” *National Institute of Standards and Technology, Federal Information Processing Standards*, p. 29, 1985.

[204] D. M’Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, “RFC 4226: HOTP: An HMAC-based one-time password algorithm,” *Internet Engineering Task Force (IETF)*, 2005.

[205] D. M'Raihi, S. Machani, M. Pei, and J. Rydell, "RFC 6238 TOTP: Time-Based One-Time Password Algorithm," *tools.ietf.org*, 2011.

[206] "Google authenticator PAM module." [Online]. Available: https://github.com/google/google-authenticator.

[207] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[208] "Playstation Blog website." [Online]. Available: https://blog.playstation.com/archive/2011/04/28/playstation-network-and-qriocity-outage-faq/.

[209] "Password Hashing Competion website." [Online]. Available: https://password-hashing.net/.

[210] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: New generation of memory-hard functions for password hashing and other applications," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016, pp. 292–302.

[211] S. Josefsson, "PKCS# 5: Password-based key derivation function 2 (pbkdf2) test vectors," *Internet Engineering Task Force (IETF), RFC Editor, RFC*, vol. 6070, 2011.

[212] "LastPass Security." [Online]. Available: https://www.lastpass.com/enterprise/security.

[213] C. Forler, S. Lucks, and J. Wenzel, "Catena: A memory-consuming password-scrambling framework," Citeseer, 2013.

[214] "PBKDF2 (Encyclopedia - Science Tech Computing)," 03-Feb-2022. [Online]. Available: https://handwiki.org/wiki/PBKDF2.

[215] "Smartphone Forensics." [Online]. Available: https://blog.elcomsoft.com/2010/09/smartphone-forensics-cracking-blackberry-backup-passwords/.

[216] "Lastpass Security Notification." [Online]. Available: https://blog.lastpass.com/2011/05/lastpass-security-notification/.

[217] M. S. Turan, E. B. Barker, W. E. Burr, and L. Chen, "Sp 800-132. Recommendation for password-based key derivation: Part 1: Storage applications." National Institute of Standards & Technology, 2010.

[218] "Argon 2 website." [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-cfrg-argon2/.

[219] M. A. Simplicio Jr., L. C. Almeida, E. R. Andrade, P. C. F. dos Santos, and P. S. L. M. Barreto, "Lyra2: Efficient Password Hashing with High Security against Time-Memory Trade-Offs." Cryptology ePrint Archive, Report 2015/136, 2015.

[220] C. Forler, E. List, S. Lucks, and J. Wenzel, "Overview of the candidates for the password hashing competition," in *International Conference on Passwords*, 2014, pp. 3–18.

[221] T. Pornin, "The MAKWA password hashing function." April, 2015.

[222] B. A. Huberman, M. Franklin, and T. Hogg, "Enhancing privacy and trust in electronic communities," in *Proceedings of the 1st ACM conference on Electronic commerce*, 1999, pp. 78–86.

[223] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private Set Intersection using Permutation-based Hashing," in *24th USENIX Security Symposium*, 2015, pp. 515–530.

[224] D. Boneh, E.-J. Goh, and K. Nissim, “Evaluating 2-DNF formulas on ciphertexts,” in *Theory of Cryptography Conference*, 2005, pp. 325–341.

[225] J. Li, N. Li, and R. Xue, “Universal accumulators with efficient nonmembership proofs,” pp. 253–269, 2007.

[226] R. C. Merkle, “A digital signature based on a conventional encryption function,” in *Conference on the theory and application of cryptographic techniques*, 1987, pp. 369–378.

[227] E. Morais, C. van Wijk, and T. Koens, “Zero Knowledge Set Membership,” 2018.

[228] “Bitwise operation,” 03-Feb-2022. [Online]. Available: https://en.wikipedia.org/wiki?curid=264399.

[229] C. Hazay and T. Toft, “Computationally secure pattern matching in the presence of malicious adversaries,” in *International Conference on the Theory and Application of Cryptology and Information Security*, 2010, pp. 195–212.

[230] J. Plass and S. Zickau, “PARADISE - Wie Ortungstechnologien den Datenschutz im Anti-Doping verbessern können,” in *Kritik des Anti-Doping - Eine konstruktive Auseinandersetzung zu Methoden und Strategien im Kampf gegen Doping*, Nils Zurawski, Marcel Scharf, 2019, pp. 131–157.

[231] “The Information website.” [Online]. Available: https://www.theinformation.com/articles/developers-call-apple-privacy-changes-anti-competitive.

[232] “Skyhook website.” [Online]. Available: https://www.skyhook.com/.

[233] “Wigle website.” [Online]. Available: https://wigle.net/.

[234] “Radio cells website.” [Online]. Available: https://radiocells.org/.

[235] “Opencell-ID website.” [Online]. Available: https://opencellid.org/.

[236] “ProgrammableWeb Geomena API website.” [Online]. Available: https://www.programmableweb.com/api/geomena.

[237] “Pequin project – Github page.” [Online]. Available: https://github.com/pepper-project/pequin.

[238] M. Keller, “A Python 3 library implementing the Paillier Partially Homomorphic Encryption,” 20-Apr-2018. [Online]. Available: https://github.com/data61/python-paillier.

[239] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *International conference on the theory and applications of cryptographic techniques*, 1999, pp. 223–238.

[240] J. Eberhardt and S. Tai, “Zokrates-scalable privacy-preserving off-chain computations,” in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1084–1091.

[241] E. Morais, “Zero Knowledge Range Proofs and Set Membership.” [Online]. Available: https://github.com/ing-bank/zkrp.

[242] “The Pepper Project - toward practical verifiable computation,” 16-Jan-2022. [Online]. Available: https://www.pepper-project.org/.

[243] I. Denisow, “JCPABE.” [Online]. Available: https://github.com/TU-Berlin-SNET/JCPABE.

[244] J. Bethencourt, “Cpabe.” [Online]. Available: http://hms.isi.jhu.edu/acsc/cpabe/.

[245] “Nominatim (An OpenStreetMap Service).” [Online]. Available: https://nominatim.openstreetmap.org/ui/search.html.

[246] “Django Framework.” [Online]. Available: http://www.django-rest-framework.org/.

[247] “Green Unicorn load balancer.” [Online]. Available: http://gunicorn.org/.

[248] “Google Firebase Cloud Messaging.” [Online]. Available: https://firebase.google.com/docs/cloud-messaging/.

[249] “GeoJSON Online.” [Online]. Available: http://geojson.org/.

[250] M. Schwochow, “Postleitzahlen Deutschland,” 2021. [Online]. Available: https://www.suche-postleitzahl.org/.

[251] “Github Germany GeoJSON.” [Online]. Available: https://github.com/isellsoap/deutschlandGeoJSON.

[252] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub, and others, “The geojson format,” *Internet Engineering Task Force (IETF)*, 2016.

[253] Bundesverband CarSharing e.V., Ed., “Carsharing Facts Germany.” [Online]. Available: https://carsharing.de/alles-ueber-carsharing/carsharing-zahlen/aktuelle-zahlen-fakten-zum-carsharing-deutschland.

[254] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, “From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again,” in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 2012, pp. 326–349.

[255] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, “Bulletproofs: Short proofs for confidential transactions and more,” in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 315–334.

[256] S. Zickau, D. Thatmann, A. Butyrtschik, I. Denisow, and A. Küpper, “Applied Attribute-based Encryption Schemes,” in *Proceedings of the 19th IEEE International Conference Innovation in Clouds, Internet and Networks (ICIN 2016)*, 2016, pp. 88–95.

[257] Z. Liu and D. S. Wong, “Practical Attribute Based Encryption: Traitor Tracing, Revocation, and Large Universe,” *IACR Cryptology ePrint Archive*, 2014.

[258] B. Lynn, “On the implementation of pairing-based cryptosystems,” PhD thesis, Stanford University, 2007.

[259] “OpenSSL.” [Online]. Available: https://www.openssl.org/.

[260] M. Green, J. A. Akinyele, and M. Rushanan, “libfenc: The Functional Encryption library.” [Online]. Available: http://code.google.com/p/libfenc.

[261] J. A. Akinyele, M. W. Pagano, M. D. Green, C. U. Lehmann, Z. N. Peterson, and A. D. Rubin, "Securing electronic medical records using attribute-based encryption on mobile devices," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 2011, pp. 75–86.

[262] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.

[263] A. Lewko, A. Sahai, and B. Waters, "Revocation Systems with Very Small Private Keys." Cryptology ePrint Archive, Report 2008/309, 2008.

[264] B. Waters, "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization," in *Public Key Cryptography – PKC 2011*, vol. 6571, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds. Springer Berlin Heidelberg, 2011, pp. 53–70.

[265] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the Internet of Things," *Future Generation Computer Systems*, vol. 49, no. 0, pp. 104–112, 2015.

[266] Y. Rouselakis and B. Waters, "New Constructions and Proof Methods for Large Universe Attribute-Based Encryption." Cryptology ePrint Archive, Report 2012/583, 2012.

[267] T. Okamoto and K. Takashima, "Fully Secure Unbounded Inner-Product and Attribute-Based Encryption." Cryptology ePrint Archive, Report 2012/671, 2012.

[268] A. Lewko and B. Waters, "New Proof Methods for Attribute-Based Encryption: Achieving Full Security through Selective Techniques." Cryptology ePrint Archive, Report 2012/326, 2012.

[269] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro, "A Framework and Compact Constructions for Non-monotonic Attribute-Based Encryption," in *Public-Key Cryptography – PKC 2014*, vol. 8383, H. Krawczyk, Ed. Springer Berlin Heidelberg, 2014, pp. 275–292.

[270] A. Lewko and B. Waters, "Decentralizing Attribute-Based Encryption," in *Advances in Cryptology - EUROCRYPT 2011*, vol. 6632, KennethG. Paterson, Ed. Springer Berlin, 2011, pp. 568–588.

[271] Y. Rouselakis and B. Waters, "Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption," in *Financial Cryptography and Data Security*, Springer Berlin, 2015, pp. 315–332.

[272] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems," *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 11, pp. 1790–1801, Nov. 2013.

[273] K. Yang and X. Jia, "Expressive, Efficient, and Revocable Data Access Control for Multi-Authority Cloud Storage," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 7, pp. 1735–1744, 2014.

[274] Q. Liu, G. Wang, and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Information Sciences*, vol. 258, no. 0, pp. 355–370, 2014.

[275] J. de Muijnck-Hughes, "pyPebel," 2013. [Online]. Available: https://github.com/jfdm/pyPEBEL.

[276] B. Lynn, “PBC library the pairing-based cryptography library.” [Online]. Available: https://crypto.stanford.edu/pbc/.

[277] “MIRACL.” [Online]. Available: https://github.com/CertiVox/MIRACL.

[278] “RELIC.” [Online]. Available: https://github.com/relic-toolkit/relic.

[279] J. Wang, “cpabe (Java),” 2013. [Online]. Available: https://github.com/junwei-wang/cpabe.

[280] A. De Caro and V. Iovino, “jPBC: Java pairing based cryptography,” in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, 2011, pp. 850–855.

[281] A. Butyrtschik, “jTR-ABE.” [Online]. Available: https://github.com/TU-Berlin-SNET/jTR-ABE.

[282] Y. Zheng, “KPABE.” [Online]. Available: https://github.com/gustybear/kpabe.

[283] Y. Zheng, “Privacy-preserving personal health record system using attribute-based encryption,” PhD thesis, Worcester Polytechnic Institute, 2011.

[284] S. Braghin, “DCPABE.” [Online]. Available: https://github.com/stefano81/dcpabe.

[285] M. Morales-Sandoval and A. Diaz-Perez, “DET-ABE: A Java API for Data Confidentiality and Fine-Grained Access Control from Attribute Based Encryption,” in *Information Security Theory and Practice*, Springer International Publishing, 2015, pp. 104–119.

[286] S. Jahid, “PIRATTE,” 2013. [Online]. Available: https://bitbucket.org/hatswitch/piratte.

[287] A. D. Caro, “Arcanum.” [Online]. Available: https://github.com/adecaro/arcanum.

[288] S. Gorbunov, V. Vaikuntanathan, and H. Wee, “Attribute-based Encryption for Circuits,” in *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, 2013, pp. 545–554.

[289] C. Gentry, S. Gorbunov, S. Halevi, V. Vaikuntanathan, and D. Vinayagamurthy, “How to Compress (Reusable) Garbled Circuits.” Cryptology ePrint Archive, Report 2013/687, 2013.

[290] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters, “Attribute-Based Encryption for Circuits from Multilinear Maps.” Cryptology ePrint Archive, Report 2013/128, 2013.

[291] D. Boneh, V. Nikolaenko, and G. Segev, “Attribute-Based Encryption for Arithmetic Circuits.” Cryptology ePrint Archive, Report 2013/669, 2013.

[292] E. Zavattoni, “LSSS2.” [Online]. Available: https://bitbucket.org/herumi/lsss2/src.

[293] E. Zavattoni, L. J. Dominguez Perez, S. Mitsunari, A. H. Sanchez-Ramirez, T. Teruya, and F. Rodriguez-Henriquez, “Software Implementation of an Attribute-Based Encryption Scheme,” *Computers, IEEE Transactions on*, vol. 64, no. 5, pp. 1429–1441, May 2015.

[294] A. H. Sanchez and F. Rodriguez-Henriquez, “NEON Implementation of an Attribute-Based Encryption Scheme,” in *Applied Cryptography and Network Security*, vol. 7954, M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds. Springer Berlin Heidelberg, 2013, pp. 322–338.

[295] M. Ambrosin, M. Conti, and T. Dargahi, “AndrABEn.” 2014.

[296] M. Ambrosin, M. Conti, and T. Dargahi, “On the Feasibility of Attribute-Based Encryption on Smartphone Devices,” *arXiv preprint arXiv:1504.00619*, 2015.

[297] E. Elmasllari and J. Plass, “Domain and requirements for a wearable-based doping control system,” *Datenschutz und Datensicherheit-DuD*, vol. 41, no. 12, pp. 717–720, 2017.

[298] C. Putschli, “Wearables und Datenschutz,” *Datenschutz und Datensicherheit-DuD*, vol. 41, no. 12, pp. 721–723, 2017.

[299] Unterarbeitsgruppe „Standard Data Protection Model“ of the AK Technik of the Independent Data Protection Supervisory Authorities of the Federation and the Länder, Ed., “The Standard Data Protection Model, Version 2.0b,” 17-Apr-2020. [Online]. Available: https://www.datenschutzzentrum.de/uploads/sdm/SDM-Methodology_V2.0b.pdf.

[300] H. A. Jäger, L. Abdullah, and J. Quintero, “Vertrauenswürdiges Backend,” *Datenschutz und Datensicherheit-DuD*, vol. 41, no. 12, pp. 729–734, 2017.

[301] A. Schlarmann, “Datenschutz beim Kampf gegen Doping. Meldepflichten deutscherAthleten nach dem neuen Anti-Doping-Gesetz,” *Zeitschrift für Datenschutz*, Dec. 2016.

[302] M. Freese, “Sprinter jagt Dopingsündern hinterher, NWZ Online,” 17-Oct-2014. [Online]. Available: http://www.nwzonline.de/handball/sprinter-jagt-dopingsuendern-hinterher_a_19,0,2264467928.html.

[303] “CATLES Service.” [Online]. Available: http://ubicom.snet.tu-berlin.de/catles/.

[304] F. Victor, S. R. Garzon, and A. Küpper, “Smartphone-collected mobile network events for mobility modeling,” in *Proceedings of the 14th IEEE International Conference on Ubiquitous Intelligence and Computing*, 2017, pp. 871–878.

[305] A. Uzun, L. Lehmann, T. Geismar, and A. Küpper, “Turning the OpenMobileNetwork into a Live Crowdsourcing Platform for Semantic Context-aware Services,” in *Proceedings of the 9th International Conference on Semantic Systems*, 2013, pp. 89–96.

[306] A. Uzun, “Linked Crowdsourced Data - Enabling Location Analytics in the Linking Open Data Cloud,” in *Proceedings of the IEEE 9th International Conference on Semantic Computing*, 2015, pp. 40–48.

[307] “Android Location Developer API.” [Online]. Available: https://developer.android.com/reference/android/location/Location.

[308] Z. A. Lux, F. Beierle, S. Zickau, and S. Göndör, “Full-text Search for Verifiable Credential Schemas on Distributed Ledgers,” in *Proceedings of the International Symposium on Blockchain Computing and Applications (BCCA 2019)*, 2019.

[309] Z. A. Lux, D. Thatmann, S. Zickau, and F. Beierle, “Distributed-ledger-based authentication with decentralized identifiers and verifiable credentials,” in *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, 2020, pp. 71–78.

[310] S. Rodriguez Garzon and B. Deva, “Sensafety: Crowdsourcing the Urban Sense of Safety,” *Advances in Cartography and GIScience of the ICA*, 2019.

[311] M. Salem and B. Deva, “3rd Party Geolocation Messaging: A Positioning Enabler Middleware for Realizing Context-aware Polling,” in *Proceedings of the 6th International Conference on Mobile Wireless Middleware, Operating Systems, and Applications*, 2013, pp. 100–109.

[312] P. Ruppel, C. Klein, and C. Linnhoff-Popien, “Indooria – A Platform for Proactive Indoor Location-based Services,” in *IEEE Global Communications Conference - Design & Developers Forum*, 2008.

[313] P. Ruppel and F. Gschwandtner, “Spontaneous and Privacy-friendly Mobile Indoor Routing and Navigation,” in *Proceedings of the 2nd Workshop on Services, Platforms, Innovations and Research for new Infrastructures in Telecommunications (SPIRIT 2009)*, 2009.

[314] E. Klarreich, “Computer Scientists Achieve ‘Crown Jewel’ of Cryptography,” 10-Nov-2020. [Online]. Available: https://www.quantamagazine.org/computer-scientists-achieve-crown-jewel-of-cryptography-20201110/.

[315] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and N. Trieu, “Practical multi-party private set intersection from symmetric-key techniques,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1257–1272.

[316] M. Naor, Y. Naor, and O. Reingold, “Applied Kid Cryptography or How to convince your children you are not cheating,” *Journal of Craptology*, 1998.

[317] M. Lodder, “ZKP Webinar: Community Call- Shared screen with speaker view, website.” [Online]. Available: https://zoom.us/rec/play/v5Z-cuD7_203S9OTsASDB6R6W424e62shClK__Ncyx7hUHIKNAWkZrcbNuPf4Kh660kl-d3dLrFHDgTR.

[318] C.-P. Schnorr, “Efficient identification and signatures for smart cards,” in *Conference on the Theory and Application of Cryptology*, 1989, pp. 239–252.

[319] D. M. Freeman, “Homomorphic encryption and the BGN Cryptosystem.” Citeseer, 2011.

[320] V. S. Miller, “The Weil pairing, and its efficient calculation,” *Journal of cryptology*, vol. 17, no. 4, pp. 235–261, 2004.

[321] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, “Efficient batched oblivious PRF with applications to private set intersection,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 818–829.

[322] A. Fiat and A. Shamir, “How to prove yourself: Practical solutions to identification and signature problems,” in *Conference on the theory and application of cryptographic techniques*, 1986, pp. 186–194.

[323] M. Slawik, S. Zickau, D. Thatmann, J. Repschläger, T. Ermakova, A. Küpper, and R. Zarnekow, “Innovative Architektur für sicheres Cloud Computing: Beispiel eines Cloud-Ecosystems im Gesundheitswesen,” in *Proceedings of the 42th Annual Conference of the Gesellschaft für Informatik e.V. (INFORMATIK 2012)*, 2012, pp. 1075–1082.

[324] D. Thatmann, M. Slawik, S. Zickau, and A. Küpper, “Towards a Federated Cloud Ecosystem: Enabling Managed Cloud Service Consumption,” in *Economics of Grids, Clouds, Systems, and Services, GECON 2012*, 2012, vol. 7714, pp. 223–233.

[325] D. Thatmann, M. Slawik, S. Zickau, and A. Küpper, “Deriving a Distributed Cloud Proxy Architecture for Managed Cloud Service Consumption,” in *Proceedings of the 6th Intl. Conference on Cloud Computing (CLOUD 2013)*, 2013, pp. 614–620.

[326] T. Graf, S. Zickau, and A. Küpper, “Enabling Location-based Services on Stationary Devices using Smartphone Capabilities,” in *Mobile Web Information Systems*, 2013, vol. 8093, pp. 49–63.