DOCTORAL SCHOOL
MEDITERRANEA UNIVERSITY OF REGGIO CALABRIA

DEPARTMENT OF INFORMATION ENGINEERING, INFRASTRUCTURES
AND SUSTAINABLE ENERGY
(DIIES)

PHD IN
INFORMATION ENGINEERING

S.S.D. ING-INF/05
XXXV CYCLE

PROXIMITY-BASED SERVICES: PRIVACY AND SECURITY ISSUES

CANDIDATE
CECILIA LABRINI

ADVISOR
Prof. FRANCESCO BUCCAFURRI

COORDINATOR
Prof. ANTONIO IERA

REGGIO CALABRIA, JANUARY 2023

Finito di stampare nel mese di **Gennaio 2023**

Edizione CSdA Centro Stampa d'Ateneo

Quaderno N. 50

Collana *Quaderni del Dottorato di Ricerca in Ingegneria dell'Informazione*
Curatore *Prof. Antonio Iera*

ISBN

Università degli Studi *Mediterranea* di Reggio Calabria
Via dell'Università 25. Reggio Calabria

CECILIA LABRINI

PROXIMITY-BASED SERVICES: PRIVACY AND SECURITY ISSUES

Contents

List of Figures

List of Tables

1 Introduction

1.1 Scenario and Motivation

The information related to the proximity between people or between people and objects and the related processing of this information are taking on a very important role, clearly amplified by the possibility of having digital tools. In fact, the rapid developments of mobile Internet, localization technologies, and social networks have made it possible to realize proximity-based services.

The concept of proximity always refers to the presence of humans, and in particular to the proximity between them or with respect to a specific target. Since sensitive information concerns the people involved, severe privacy and security issues arise. Therefore, this thesis focuses precisely on the problem of privacy and security.

However, proximity can have different meanings and the thesis aims to address and analyze the different meanings of proximity and the different contexts.

The first meaning is that of proximity between people for social reasons. According to the definition provided by Aristotle, *"Man is by nature a social animal"*. This feature has always characterized the human being and has found new forms of expression in this modern and technological era. Indeed, the rapid and widespread development of social networks is explained precisely by this need to enter into a relationship with one's peers. Social networks probably represent the most disrupting digital innovation of the last twenty years.

Different kinds of applications are nowadays implemented on top of social networks. Among these, the services that are achieving more popularity are those that allow finding friends nearby, or even unknown people or nearby points of interest. In fact, the major social networks provide their users with proximity-based services, such as the *Nearby Friends* feature of Facebook. There are also social networks founded on geospatial features, among which, services based on the reciprocal proximity of users like Foursquare, Jiepang, FullCircle, Tinder, Gowalla, and Facebook Places. For example, Tinder allows unknown users can come into contact if they are in proximity and match some preferences. Specifically, each user advertises a set of attributes (possibly, not identifying) and expresses a preference for the attributes of other users. To detect proximity, it is necessary that the users reciprocally show an expression of interest. Another possible service regards proximity testing performed by users with respect to a given target (static or moving). In the context of car sharing, the targets are the available cars of a given company. Other examples of moving targets are ride-sharing, crowd-shipping, proximity marketing, or applications supporting people to obtain aids by volunteers when they are away (consider for example the case of blind people). Instead, an example of a static target is represented by points of interest (POIs).

In these services, a relevant issue from the side of privacy is that we do not have guarantees that the provider is fully trusted and enough immune to data breaches. In the literature, many research works are concentrating their efforts on this field of proximity-based services [111, 98, 120, 128, 85], but never in the most severe threat model of a global passive adversary. Concretely, this is the case of services entirely delivered to social-network users within the social network (due to the fact that the social network provider, playing as a global passive adversary, can monitor the flow of all the messages in the network), without assuming external communication channels.

The *first research question* we address in this thesis is: *"How can we provide proximity-based services within social networks, guaranteeing the privacy of the users involved against a global passive adversary?"* Therefore, we deal with this problem and propose a solution allowing any pair of users to perform a *proximity test* without revealing to the adversary the fact that the test is performed. This goal has not been reached previously in the literature. In particular, we provide three proximity-based services. The first is a service aimed to test the proximity of users who know each other. Roughly, we provide the privacy features to a service similar to *Facebook Nearby Friends*. The second service is used to test the proximity between users who do not know each other but make public some information (photos, preferences, etc.). The service allows detecting proximity of unknown users only on the basis of their agreement. This service extends the features given by services such as *Tinder*, by enabling the above privacy features. Finally, the last service regards proximity testing of a user with respect to a (static or moving) target. The privacy requirement is that the user remains anonymous also with respect to the target. This service extends services such as *Tripadvisor* or *BlaBlaCar*.

Social proximity has since ever been evaluated as positive. The pandemic emergency of recent years attributed a second meaning to the concept of proximity which is proximity detection for contact tracing. The concept of proximity between people has never been more important. The outbreak of the COVID-19 pandemic has dramatically reduced our social relations to avoid spreading the contagion. The COVID-19 pandemic is one of the most difficult challenges that modern society has ever faced. To counter and slow down the spread of the virus, new ways, new strategies, and solutions are being sought every day, in every sector, from the economic to the medical, from the political to the technological. Precisely, in the latter field, researchers are investing their effort to propose, digital solutions for contact tracing that preserve privacy and comply with current regulations. Digital contact tracing (DCT) should be considered as a complementary task with respect to traditional contact tracing because it is able to identify contacts that escape the investigation activities carried out by contact tracers (for example, whether they regard contacts with people unknown to the index case). In addition, the specific characteristics of COVID-19 infection (variable symptoms, frequent asymptomatic carriers, and incubation times relatively short) require faster detection of at-risk contacts than traditional contact tracing. DCT represents one of the weapons that information technology can provide to fight the pandemic.

In the European Union, the prevailing protocol is DP-3T (Decentralized Privacy-Preserving Proximity Tracing) [166], typically implemented via GAEN (Google-Apple Exposure Notification) [20]. DP-3T/GAEN is decentralized (i.e., does not delegate contact detection to a server) and does not utilize localization systems (such as GPS) to detect proximity, but only BLE. A number of vulnerabilities have been reported about DP-3T/GAEN [23, 170, 26], which can lead to break protocol integrity and users' privacy.

Therefore, the *second research question* that we address in this thesis is: *"Can we realize an alternative approach that overcomes the most drawbacks of DP-3T/GAEN?"*

It is not only pandemic emergencies that cause numerous victims or force us to live in the worry of coming into contact with an invisible and lethal enemy, but there are also other emergencies, of which we talk too little, which cause numerous victims every year, mainly young women, all over the world. Therefore, a third meaning that we deal with in this thesis is proximity in the domain of controlling people's safety. The number of victims of stalking is constantly increasing and too often the solutions put in place turn out to be inadequate. Also, in this case, proximity-based services can represent an easy and safe solution. Electronic monitoring is a valuable approach to the control of sex offenders. It also avoids prison overcrowding and protects victims. Two technologies are currently being adopted: RFID and GPS. GPS is the best choice when high-security requirements are desired. In fact, radio frequency attacks are possible for RFID, endangering the victim. However, when GPS is adopted, privacy issues become critical.

In this thesis, we analyze the problem of EM in the general case, by identifying a gap in existing techniques in the case of dynamic exclusion zones, which is the most critical case from the victim's privacy perspective. Therefore, the *third research question* that we address is: *"Can we realize a solution for electronic monitoring to preserve the privacy of the victim?"*

In all these proximity cases, the way in which geolocation information is managed clearly plays an important role. It is also necessary to manage the territory in order to efficiently process this information. Therefore, the *fourth research question* that we address in this thesis is: *"Can we design an efficient representation capable of supporting proximity detection at different ranges?"* We adopt a tag-grid-based approach that consists of the partition of the territory into *cells* of a certain shape (squares, hexagons, circles, etc.), possibly overlapping each other. We design a hierarchical spatial index based on the concept of *quad tree*, in which also overlapping is enabled. We call this structure *shifted quad tree* (SQT).

Furthermore, often these data are in large quantities and therefore there is a need to manage them in outsourcing. The proximity service provider outsources the map data to a third party (typically the cloud), which however may not be honest. The proximity service provider needs guarantees on the completeness and correctness of the portion of map data returned. Therefore, the *fifth research question* that we address in this thesis is: *"Can we implement an approach able to guarantee query integrity over map data outsourced to a cloud?"* To answer this question, we implemented a lightweight message-authentication-code-based approach.

1.2 Outline of the Thesis

This thesis proposes new solutions to realize proximity-based services guaranteeing privacy. Proximity-based services expose the user to serious privacy threats because they could allow massive monitoring by an honest but curious provider. We propose several protocols to implement proximity-based services in different contexts, such as social networks, electronic proximity monitoring for the prevention of crimes, and contact tracing, always with the main objective of ensuring privacy.

For each proposed solution, a security analysis is performed.

The thesis is organized as follows:

In Chapter 2, we provide an overview of the proximity-based services found in the literature. These services are classified into four main groups: *grid-based*, *tag-based*, *location-anonymity-based*, *encryption-based*. Furthermore, this chapter provides some background notions useful to better understand the protocols proposed in the thesis.

Chapter 3 introduces some fundamental concepts and solutions that constitute the starting point for realizing the proximity-based services presented in the following chapters. In particular, this chapter focuses on the management of the mapping of the territory. Therefore, we propose an efficient representation capable of supporting proximity detection at different ranges. To achieve this goal, we design a grid-based solution that supports the modulation of the range of action of proximity testing. Thus, we propose a new hierarchical spatial index, called *shifted quad tree*, allowing the user to choose the distance within which proximity testing is performed. Furthermore, we concentrate on query integrity over map data outsourced to a (possibly dishonest) cloud. We propose an approach based on a flat data structure, opposite to tree-like of the state-of-the-art solutions. The solution advances the state of the art.

In Chapter 4, using the solutions we have designed in Chapter 3, we propose a solution that allows the realization of proximity-based services in Social Networks guaranteeing privacy. Therefore, after defining an anonymity protocol and communication primitives, we develop a privacy-preserving proximity-based solution that provides both symmetric and asymmetric proximity testing entirely within social networks. We realize three different proximity-based services: KN-service (Proximity between known users), UN-service (Proximity between unknown users), and TN-service (Proximity between an anonymous user and a target). The services are implemented through several phases that guarantee the privacy of the users involved.

In Chapter 5, we propose an approach for electronic proximity monitoring of sex offenders, and more generally, for the prevention of crimes. We focus on this topic from a perspective that considers both the privacy of the victim and the security of the solution against the offender's misbehavior. In this context, Radio Frequency (RF) and Global Positioning System (GPS) are the reference technologies but radio frequency attacks are possible for RF, endangering the victim, and instead, when GPS is adopted, privacy issues become critical. To overcome this drawback, we propose a GPS-based solution that does not allow the victim's location to be revealed unless the offender is nearby, thus finding a solution that advances the state of the art.

Chapter 6 focuses on the topic of digital contact tracing (DCT), in particular proximity tracing, as an important, effective, and privacy-preserving measure for curbing the spread of pandemic disease outbreaks. We propose a new centralized DCT protocol. Unlike DP-3T/GAEN, the proposed protocol does not rely on the exchange of ephemeral identities among users. Furthermore, our solution does not use Bluetooth, therefore users are not exposed to existing Bluetooth vulnerabilities.

We prove that ZE2-P3T is more secure than the state-of-the-art approach, i.e., DP-3T/GAEN.

2

Approaches for Proximity-Based Services

This chapter offers an overview of the most significant proximity-based services, with the aim of framing the research field in which the thesis moves and showing the currently existing solutions. A classification of proximity-based services approaches is provided. Furthermore, this chapter provides some background notions useful to better understand the protocols proposed in the thesis. The concepts given in this chapter form a basis for the development of the protocols presented in subsequent chapters.

2.1 Introduction

Thanks to advances in positioning technologies and to the spread of mobile devices with data communication capabilities, location-based services (LBS) are becoming popular. Proximity-based services are a special class of LBS in which service adaptation depends on the comparison between a given threshold value and the distance between a user and other (possibly moving) entities [111]. Proximity services are becoming more and more important, just think of meeting apps, contact tracing, proximity advertising, national security, proximity marketing, etc. The pandemic emergency brought out the importance of the concept of proximity between people. Digital contact tracing (DCT), in particular proximity tracing, is an important, effective, and privacy-protecting measure to stem the spread of pandemic disease outbreaks [144]. Researchers are investing their effort to propose, digital solutions for tracing contacts that preserve privacy and that comply with current regulations (as shown in detail in Section 6.2). Furthermore, in the field of the Industrial Internet of Things (IIoT), mechanisms have been proposed that exploit the concept of proximity [139]. Apple's Bluetooth Low Energy (BLE), named *iBeacon*, it is a widely used technology in the Proximity-based Services domain. However, it has several limitations. It suffers from poor proximity detection accuracy due to its reliance on Received Signal Strength Indicator (RSSI). [184] offers an improved solution.

Social networks probably represent the most disrupting digital innovation of the last twenty years. Different kinds of applications are nowadays implemented on top of social networks. The power of social networks could be better exploited in various application contexts, such as e-democracy, e-participation, online surveys, crowdsourcing, proximity-based services, and so on. Current social networks offer proximity-based services. For example, Facebook supports the meeting feature *Nearby Friends*. There are also social networks founded on geospatial features, providing services based on the reciprocal proximity of users like Foursquare, Jiepang, FullCircle, Tinder, Gowalla, and Facebook Places. Nowadays, more and more users use the proximity detection service such as "people in the vicinity", and "nearby restaurants", which make the proximity detection becomes a basic service in mobile social networks. A well-known example of these services is the "friend-finder" service, in which a user, for example, Alice, would like to be notified whenever her friend, for example, Bob, is nearby so they could get in contact and eventually meet.

The traditional proximity detection methods require users to submit their location information to the location server in order to find their neighbors. Technically, a proximity query is a spatial range query on the database in which the range is defined by the circle centered at the issuer's location and having the proximity threshold as radius.

Unfortunately, in these services, users' privacy is threatened. A major privacy concern with the use of LBS is the release to untrusted third parties of the user precise location information. Indeed, an *honest-but-curious* (also said *semi-trusted*) service provider could misuse location data, which are potentially sensitive. In the recent literature [141], there is growing attention toward proximity-based social networking. The problem of privacy in proximity-based services has been deeply analyzed in the literature for more than a decade [111, 98, 120, 128, 85]. Note that the possible threats may differ, for example, if the location server has a security vulnerability or the insider abuses users' location information, the location privacy of users will be disclosed [181]. It could also happen that Alice wants to use the proximity service without necessarily releasing her exact position to the service provider (SP). Another possible circumstance is that Alice does not want to give the exact position to her friends, although she may be willing to reveal whether she is in proximity. For example, she may agree to let Bob know that she is in a neighborhood near Bob's location, but keep the specific address hidden from Bob. This may avoid the situation in which friends can directly walk to other friends, as the goal of the service is usually to enable communication that may only eventually lead to meetings in person. A solution to the above privacy concern can be to allow each user to specify certain minimum location privacy requirements both with respect to the service provider and to the friends. Observe that these are minimum requirements, and a system should be designed with the goals to (1) guarantee the satisfaction of the minimum privacy requirements, and (2) reveal as little location information as possible. [111]

Several LBS privacy preserving techniques have been recently proposed [70, 71, 89, 110, 182]. Some techniques [70, 89, 110] consider the possible use of location information contained in anonymous LBS requests to discover the identity of the issuers and hence their connection with the private information such as the specific service being requested. To guarantee a given level of anonymity of the users, different spatial generalization functions are proposed. An anonymized request may contain a quite precise location, since, independently from the actual size of the region, a generalization is considered satisfactory whenever the region contains a sufficiently large number of potential issuers.

These techniques perform in many scenarios but are not adequate for proximity services since location is considered private information. There are techniques more focused on the obfuscation of private information and in particular on the user's location. A possible example is SpaceTwist [182], which is specifically designed for K-NN queries and hides the location of the user by issuing a sequence of requests reporting fake locations. Other solutions, as [71], exploit private information retrieval (PIR) techniques to encrypt all of the information exchanged with the service provider and to process the corresponding query in an encrypted form so that no location information is revealed to the SP. In [91], the use of PIR techniques has been proposed to calculate range queries between static resources. However, it is not clear how these methods can be extended to computationally feasible solutions to handle range queries in which friends, unlike fixed resources, are moving around. Otherwise, [111] considers dynamic groups. The problem of privacy in proximity-based services has been deeply analyzed in the literature but never in the most severe threat model of a global passive adversary.

2.2 Proximity-Based Services Classification

The techniques for *privacy-preserving proximity testing* fall within the more general class of location-based services (LBS) [86]. To preserve location privacy in Location Based Services (LBS), several solutions have been designed. The most significant are the following.

To hide all users' locations, [32] proposed a conception of "Mix Zone". When a user enters a mix zone, he changes his pseudonym and then gets out the mix zone. It is difficult for LBS to know who the user is when the user obtains a new pseudonym and goes out of the mix zone. The concept of *k-anonymity* into location privacy was introduced in [72]. Location *k-anonymity* is satisfied when the location of a mobile user cannot be distinguished from other $k - 1$ users. Location privacy is achieved through a minimum cloaking region that contains at least k users. Instead of sending the exact location of a single user, the cloaking region is sent. This method requires the presence of a trusted third party that is responsible for generating the minimum cloaking regions by collecting the locations of different mobile users. The spatial cloaking method can be employed to reduce the accuracy of location information and protect location privacy. [21] represents an example of a spatial cloaking method, in which the user sends a circle region to LBS instead of an exact location. However, the quality of service will decline as the LBS is unable to obtain the exact locations of users.

The above methods are inadequate in the mobile social network. For example, an attacker can obtain the location information or non-location information of users. Taking advantage of this information, the attacker could reconstruct the location privacy of those users. The solutions proposed to provide a proximity detection service on the mobile social network without exposing the user's location information are several. According to [181], they can be classified into four main classes: *grid-based*, *tag-based*, *location-anonymity-based*, *encryption-based*.

2.2.1 Grid-based

The first group we consider for categorizing proximity-based services is the grid-based class. According to this approach, a large geographical area is organized as a grid, identifying (possibly overlapping) cells of a given shape. In the most common approach, a user who wants to disclose their proximity with another user, sends the service provider the cell identifier in which they are located in encrypted form (possibly by using a cryptographic hash function). In the literature, several papers follow this approach [153, 98, 154, 82, 192].

In 2009, [153] proposed a "grid-and-hashing method", which consists of dividing the space into a uniform grid unit (cell), and the ID of unit that each user located in has irreversibly hashed before sending them to the location server. Specifically, the authors developed a client-server solution for detecting proximity among friend pairs while offering them location privacy. The client maps a user's location into a grid cell, converts it into an encrypted tuple, and sends it to the server. Based on the encrypted tuples received from the users, the server determines the proximity between them blindly, without knowing their actual locations. However, this method exists false negative results as the grid division is pre-specified. This means that two mobile users close to each other, but divided into different cells, generate different hash values.

To overcome this limitation, in [98], the authors adopted the grid-and-hashing paradigm and developed two techniques, namely grid overlay and multilevel grids, to increase detection accuracy while saving wireless bandwidth. In this solution, each user has a corresponding hash vector by using a series of interlocking grids. It indicates that two users are neighbors if they have the same hash result in a particular vector dimension. Based on these techniques, they elaborated the client-side location update scheme and the server-side update handling procedure for continuous proximity detection. In this method, the layout and placing of grid will have a significant impact on reducing false negative rate.

[154] presented Vicinity Locator, a client-server solution for proximity detection, based on encrypted, multi-level partitions of the spatial domain. The service notifies a user if any friend users enter the user's specified area of interest, called the vicinity region. This region can be of any shape and can be flexibly changed on the fly. In particular, the client maps its location into a granule and finds all granules contained in his vicinity, which can be shaped arbitrarily. The client then encrypts its location- and vicinity- granules and sends them to the server, which checks for proximity by testing for the inclusion of an encrypted location granule within a set of encrypted vicinity granules of a different user.

[82] proposed a privacy-preserving proximity testing scheme, called EPPD, in which the "vicinity region" is centered on the false location generated using differential privacy techniques, which can avoid location exposure. In specific, EPPD is comprised of two phases: first, users periodically upload their encrypted locations to the service provider; and later, users can send requests to the service provider for proximity testing and obtain the final testing results.

In [192], each user can define his privacy region and two users are neighbors only when their privacy regions have an intersection. This solution enables LBS users to send proximity-test requests to the LBS server and get the proximity-test computation results from the LBS server in a privacy-preserving way. The LBS server learns nothing about the location information of the participants and the requesting user cannot learn the responding users' exact locations either. The requesting user can learn the total number of responding users and the portion of valid responses. This scheme also provides users with a method to verify the correctness of the proximity test results generated by the semi-trusted server.

2.2.2 Tag-based

This approach leverages the spatial-temporal location tags present in a specific area to allow the service provider to detect proximity. Unpredictable and unique spatial–temporal location tags can be implemented by employing a physical infrastructure or can be obtained by capturing some environmental features, such as Bluetooth IDs, WiFi IDs, LTE signals, military codes in GPS, audio signals, and atmospheric gases. An attacker cannot forge a location tag if she is not at the corresponding location and time, due to the high freshness (entropy) and spatial variety of environmental signals. The location tags can resist location cheating for malicious users, but there exist some difficulties to construct a suitable location tag. The papers [120, 188, 189, 140, 65, 181, 125, 82, 157] are good examples of this category.

[188, 189] proposed a scheme that enables a user to perform a location based handshake that establishes secure communications among strangers, who do not have a pre-shared secret, and a privacy-preserving proximity test without revealing the user's actual location to the server or other users not within the proximity. The spatial-temporal location tag is the key to proximity matching, where the fuzzy extractor is exploited to extract a secret key from two matching users. In addition, the location tag is organized in a Bloom filter, such that users can choose their own matching sensitivity with ease via tuning the parameter of the Bloom filter and BCH encoder.

In [140], the mobile network is divided into clusters. Each cluster is provided with a trusted authority. Any communication between the nodes which are located either inside or outside the cluster, takes place through the trusted authority. Each client has a public as well as a secret location tags. The system adopts a non-parametric Bayesian method known as infinite Gaussian mixture model for performing the proximity test.

[65] employs the basic geographical features of cellular networks and provides two layers of spatial anonymity such that the user's location is not directly provided to a location service provider. Based on the features of the cellular network (e.g., LTE) at the first layer, the user's location is kept hidden under the cloaking of the base station (eNB) that provides a network connectivity to serve the user (SeNB). At the second layer, the authors anonymized SeNB in a group of dummy locations neighboring a central eNB (CeNB), all of which have the same query probability.

Bluetooth low energy (BLE) beacons have been widely deployed to deliver proximity - based services to users' smartphones when users are in the proximity of a beacon. Such an approach suffers two major issues, i.e. the severe RSS fluctuation might confuse the smartphone during the detection and a malicious PBS can be delivered by manipulating the same beacon ID. [125] proposed an RF fingerprinting to label a beacon with an N-dimensional fingerprint vector, which consists of N RSS values from N deployed beacons.

In [181], a method was presented which is based on the transfer of neighbor relation. The social network server discovers neighbor relationships among users by

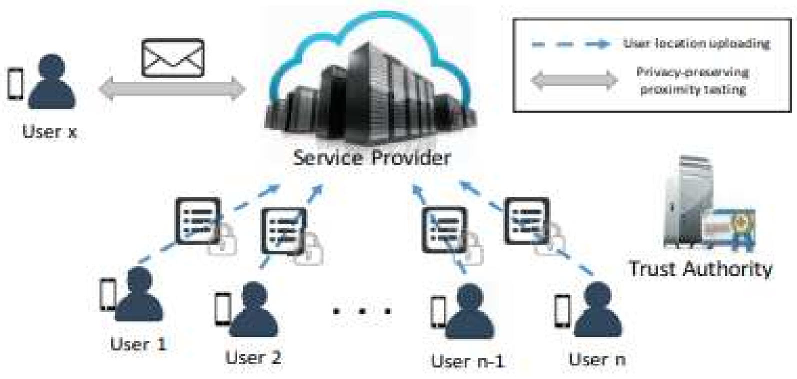


Fig. 2.1: The system model of an example of a tag-based approach.

computing users' nearby reference lists (i.e., by checking whether they have a common item). In addition, the authors also proposed another two methods—Beacon Node Rotating Mechanism and Beacon Node Competition Mechanism to determine the ways of transmitting signals of mobile nodes.

Among these papers, [82, 157, 181] are the most related to our proposal presented in Chapter 4 because they address the problem of proximity testing within social networks as our solution. However, [82, 181] achieve a privacy level less robust than our approach, since the service provider is aware of the fact that two users are performing a proximity test, thus resulting in considerable privacy leakage. Anyway, [82] requires a fully trusted authority generating users' keys, as shown in Figure 2.1. Furthermore, it is based on differential privacy, which requires much more computational effort than our approach.

Regarding [157], it is tailored for social networks but requires an external secure channel to exchange some information among the users. Therefore, it does not offer a solution fully lying within the social network.

2.2.3 Location-Anonymity-based

These techniques are based on the application of a distance-preserving transformation allowing the users to send their transformed positions to a centralized proximity detection service able to compute the reciprocal distance but not the original positions. This method is adopted by most proximity detections.

[146] well represents this class of techniques. It presented an anonymization technique for *location-based community services (LBCSs)*, which employs distance-preserving coordinate transformations in conjunction with pseudonyms. It is based on the idea that for determining the distance between targets only relative positions are needed. In practice, [146] applied a distance-preserving mapping to transform

the user's location q into a location q'. After the transformation, a centralized proximity detection method is presented to detect the proximity among the transformed locations. However, [103] pointed out that such distance-preserving mapping is not secure. An outside attacker can easily derive the secret mapping function and recover the original location of users.

[111] presented three privacy-preserving protocols for proximity services, in particular, Hide & Crypt is a method that adopts a filter-and-refine two-phase procedure. In the first phase, all users cloak their locations before sending them to the server and in the second phase, the server computes the minimum and maximum distances between these cloaking regions. The server classifies users' friends being in, not-in, or possibly-in proximity according to the specified thresholds and the computed distances.

However, the techniques based on distance-preserving transformations are prone to various vulnerabilities, as deeply studied in [109].

2.2.4 Encryption-based

The papers falling in this class of approaches face the problem mostly by using secure multi-party computation or homomorphic-based protocols [73, 80, 128, 85, 94, 112, 120]. In this case, two users can perform the privacy-preserving proximity test without revealing to each other and to the provider their position. At the end of the execution of the protocol, the provider does not know anything about the result of the proximity test.

In [190], Zhong et al. presented the Louis, Lester, and Pierre protocols for location proximity. The Louis protocol uses additively homomorphic encryption to compute the distance between Alice and Bob while it relies on a third party to perform the proximity test. Bob needs to be present online to perform the protocol. The Lester protocol does not use a third party but rather than performing proximity testing computes the actual distance between Alice and Bob. The Pierre protocol resorts to grids and leaks Bob's grid cell distance to Alice.

[120] casted the proximity testing problem as equality testing on a grid system of hexagons. One of the protocols utilizes an oblivious server. Parties in this protocol use symmetric encryption, which leads to better performance. However, this requires having preshared keys among parties, which is less amenable to one-to-many proximity testing

[73] proposed InnerCircle, a multi-party computation protocol for parallelizable decentralized proximity testing, using additively homomorphic encryption between two parties that must be online. The protocol achieves fully privacy-preserving location proximity without a trusted third party in a single round trip.

In [85], Järvinen et al. designed efficient schemes for Euclidean distance-based privacy-preserving location proximity. They demonstrate performance improvements over InnerCircle. Yet the requirement of the two parties being online applies to their setting as well.

[80] adopted a cryptographic approach to enable privacy-preserving computation of location data. Using distance computation as an example, the authors proposed methods to compute distance and perform spatial cloaking over encrypted coordinate data. Security is guaranteed by the underlying hardness problem in homomorphic encryption.

[128] proposed OLIC, a protocol for privacy-preserving proximity testing with a napping party, i.e. after providing some data about its location, one party can go offline (nap) during the proximity testing execution, without undermining user privacy.

[151] homomorphically calculated distances using the UTM projection, ECEF (Earth-Centered Earth-Fixed) coordinates, and the Haversine formula that make it possible to consider the curvature of the Earth.

2.3 Background

In this section, we provide a background on the main cryptographic techniques used in this thesis.

Cryptographic Hash Function. In order to safeguard information's authenticity, hash functions were introduced in cryptology in the late seventies. As time went on, it became obvious that they were a very useful building block for addressing other security issues in telecommunication and computer networks [138]. Hash functions are functions that compress a string of arbitrary input to a string of fixed length. If hash functions satisfy additional requirements, they are a very powerful tool in the design of techniques to protect the authenticity of the information. Therefore, hash functions can be used in a variety of cryptographic applications and thus they are known as Cryptographic Hash functions [137]. Several security goals, including authenticity, digital signatures, pseudo number generation, digital steganography, digital time stamping, etc. are achieved by using cryptographic hash functions, representing one of the most crucial tools in the science of cryptography [156]. According to Rompay [136], the formal definition of hash functions is as follows:

Definition 2.1. *A hash function is a function $h : D \to R$, where the domain $D = \{0,1\}^*$ and $R = \{0,1\}^n$ for some $n \geq 1$.*

Cryptographic Hash Functions are broad of two types: *Keyed Hash functions*, which use a secret key, and *Un-keyed Hash Functions*, which do not use a secret key.

The Keyed Hash Functions are referred to as *Message Authentication Code (MAC)*. Un-keyed Hash Functions (sometimes also known as *Manipulation Detection Code (MDC)*) can further be classified into *One Way Hash Functions(OWHF)*, *Collision Resistant Hash Functions(CRHF)*, and *Universal One way Hash Functions(UOWHF)* depending on the additional properties it satisfies.

According to [115], OWHF is a hash function H that satisfies the following requirements:
(i) H can be applied to block of data of any length.
(ii) H produces a fixed-length output.
(iii) Given H and x (any given input), it is easy to computer message digest $H(x)$.
(iv) Given H and $H(x)$, it is computationally infeasible to find x.
(v) Given H and $H(x)$, it is computationally infeasible to find x and x' such that $H(x) = H(x')$.

In order to use a hash function for message authentication and digital signatures, the first three conditions must be met. The fourth condition, often referred to as *pre-image resistance* or *one-way property*, asserts that it is easy to generate a message code given a message, but hard (virtually impossible) to generate a message given a code. The fifth condition also known as *Second pre-image resistance property* guarantees that an alternative message hashing to the same code as a given message cannot be found.

According to [114], *Collision Resistant Hash functions (CRHF)* may be defined as a Hash function H, that satisfies all the requirements of OWHF and in addition, satisfy the following collision resistance property:

Given H, it is computationally infeasible to find a pair (x, y) such that $H(x) = H(y)$.

[119] presented the idea of *Universal One Way Hash Functions (UOWHF)* and using the same, presented a digital signature scheme that was not based on trapdoor functions. [119] used 1-1 one-way functions to construct UOWHF and in turn implement the Digital Signature scheme.

Message Authentication Code (MAC) is a widely used technique to implement message authentication and integrity. Since they have been in use for much longer in the banking industry, message authentication codes predate the open research in cryptology that began in the mid-seventies. However, MACs with strong cryptographic features weren't developed until open cryptologic research got underway. Informally, a MAC is a short piece of information, associated with a message, used to assess the integrity and authenticity of the message itself.

Definition 2.2. *A MAC is a function satisfying the following conditions [137]:*
(i) The description of h must be publicly known and the only secret information lies in the key (extension of Kerckhoffs's principle).

(ii) The argument X can be of arbitrary length and the result $h(K,X)$ has a fixed length of n bits (with $n \geq 32,\ldots,64$).
(iii) Given h, X and K, the computation of $h(K,X)$ must be "easy".
(iv) Given h and X, it is "hard" to determine $h(K,X)$ with a probability of success "significantly higher" than $\frac{1}{2^n}$. Even when a large set of pairs $\{X_i, h(K,X_i)\}$ is known, where the X_i have been selected by the opponent, it is "hard" to determine the key K or to compute $h(K,X_0)$ for any $X' \neq X_i$. This last attack is called an adaptive chosen text attack.

Keyed-Hash Message Authentication Code (HMAC) is a specific kind of MAC involving a cryptographic hash function and a secret cryptographic key [95]. As with any MAC, it may be used to simultaneously verify both the data integrity and authenticity of a message. Instead of employing digital signatures with asymmetric cryptography, HMAC can enable authentication using a shared secret. HMAC trades off the need for a complex public key infrastructure by delegating the key exchange to the communicating parties, who are responsible for establishing and using a trusted channel to agree on the key prior to communication. In particular, $HMAC(K,m) = H((K \oplus opad)\|H((K \oplus ipad)\|m))$ where K is a secret key generated from a master key, m is the message to be authenticated, $\|$ denotes the operation of concatenation, $\oplus$ represents the exclusive or (XOR) operation, and $opad$ and $ipad$ are two kinds of padding, namely outer and inner padding. For both H and HMAC, the output is generically called digest.
Identity-based Encryption (IBE) is a type of public-key encryption in which the public key of a user is represented by some unique information associated with the user's identity. In IBE, each user may encrypt a message for another user without requiring the public key to any external party, by directly using the information associated with the identity of the other user.

Formally, an IBE scheme is composed of four algorithms:
Setup(k): it takes as input a security parameter k and outputs a master secret key MSK and master public key MPK.
Extract(MPK, MSK, ID): it takes as input the master public key MPK, the master secret key MSK, and a parameter ID representing the identity of a user. It outputs a private key d associated with the user's identity ID.
Encrypt(MPK, ID, M): it takes as input the master public key MPK, a parameter ID representing the identity of a user, and a message M. It outputs a ciphertext C intended for the user with identity ID.
Decrypt(MPK, C, d): it takes as input the master public key MPK, a ciphertext C, and a private key d. It outputs the decryption M of the ciphertext C.

These four algorithms are used as follows.

A trusted third party, called Private Key Generator (PKG), is involved. Preliminary, in the setup phase, the PKG invokes Setup(k) to obtain MPK and MSK. MPK is provided to all the users and MSK is kept secret by PKG. A user U with identity ID_u, who wants to obtain a secret key associated with ID_U, contacts the PKG, which invokes Extract(MPK, MSK, ID_U) to obtain d_U, and sends it to U. Clearly, the PKG sends d_U after verifying the identity of U, for example through the intervention of an Identity Provider. Suppose another user Y wants to send a message M to U (whose identity ID_U is known to Y). Y has to invoke Encrypt(MPK, ID_U, M) to obtain the ciphertext C and then can send C to U. Eventually, U invokes Decrypt(MPK, C, d_U) to retrieve the message M. Observe that Y, during the encryption process, does not interact with any party.

3

Map Data: Mapping the Territory

This chapter introduces some fundamental concepts and solutions that constitute the starting point for realizing the proximity-based services presented in the following chapters. This is due to the fact that proximity-based services are based on complex management of map data, on which the targets or the users are represented. This chapter focuses on the management of the mapping of the territory, in particular, it deals with the problem of the map data from two points of view. The first is that of efficient representation capable of supporting proximity detection at different ranges. Therefore, we propose a new hierarchical spatial index, called shifted quad tree, allowing the user to choose the distance within which proximity testing is performed. The second aspect dealt with concerns a security issue, specifically the problem of data integrity. In this case, the proximity service provider outsources the map data to a third party (typically the cloud), which however may not be honest. We propose an approach based on a flat data structure, opposite to tree-like of the state-of-the-art solutions.

3.1 Introduction

Proximity-based services are based on the complex management of map data, on which the targets of the proximity services or the users for whom the proximity must be detected are represented. In this chapter, we deal with the map problem given from two different points of view.

(i) The first is that of efficient representation capable of supporting proximity detection at different ranges. A possible scenario is represented by social networks that offer proximity-based services (such as the Nearby Friends function of Facebook). This feature exposes users to serious privacy threats because it could allow for massive monitoring by an honest but curious provider. In the literature, no solution has been provided to the problem of providing proximity services that preserve privacy entirely within existing social networks. This problem typically underlies tag-grid-based approaches. In grid-based approaches, a large geographic area is divided into cells of the same size (possibly overlapping), thus forming a grid. In the most common approach [98, 153, 154], to disclose the proximity between two users, the user sends to the service provider the cell identifier in which is located in an encryption way (possibly by using a cryptographic hash function). In tag-based approaches [120, 189, 65, 125, 82, 157], spatial-temporal location tags allow revealing proximity in a specific area in which the service is used. Unpredictable and unique tags can be implemented by employing a physical infrastructure or can be obtained by capturing some environmental features, such as Bluetooth IDs, WiFi IDs, military codes in GPS, audio signals, and atmospheric gases. [82, 157, 181] address the problem of proximity testing within social networks. However, in [82, 181], the service provider is aware of the fact that two users are performing a proximity test, thus resulting in considerable privacy leakage. [157] requires an external independent secure channel to exchange some information among the users and, therefore, it does not offer a solution fully lying within the social network. We focus on proximity-based services aiming to detect the proximity between users or between a user and a target. We have devised a tag-grid-based solution that supports the modulation of the range of action of proximity testing [40]. Therefore, we proposed a new hierarchical spatial index, called *shifted quad tree* allowing the user to choose the distance within which proximity testing is performed. We had proposed a simpler solution in [39], but it did not support the modulation of the range of action of proximity testing.

(ii) The second point of view considered is that of data integrity. Highly dynamic map data play nowadays a crucial role in different application contexts. Their huge volume often enforces the data owner to outsource them to a third party, becoming this way the client of analytical queries computed by the cloud. In this case, the proximity service provider outsources the map data to a third party (typically the cloud), which however may not be honest. The proximity service provider needs guarantees on the completeness and correctness of the portion of map data returned. Query integrity is guaranteed if three properties are fulfilled: *Completeness* (all the tuples involved in the queries have to be returned), *Correctness* (the tuples returned have not to be corrupted), and *Freshness* (the most up-to-date version of tuples has to be returned).

Since the services of interest may concern a specific area, typically the requests that the provider makes to the cloud are range queries. We proposed in [42] a lightweight message-authentication-code-based approach to guarantee query integrity over map data outsourced to a cloud. The proposed technique outperforms tree-like state-of-the-art solutions and shows the nice feature of providing guarantees for freshness without requiring timestamps, synchronization, and revocation mechanisms.

Therefore, this chapter consists of two parts: (i) The first one which focuses on managing the mapping of the hierarchical territory, using the structure called *shifted quad tree* to detect the proximity to different ranges. This structure will then be adopted in the following chapters, with the necessary modifications for the cases dealt with. (ii) The second part focuses on the integrity problem of highly dynamic outsourced map data.

3.2 Proximity Testing: Grid Organization

The proposed approach is grid-based. Usually, grid-based techniques consist of the partition of the territory into *cells* of a certain shape (squares, hexagons, circles, etc.), possibly overlapping each other. Since we want to enable the modulation of the size of the searching area in which users want to perform their proximity test, we need a more sophisticated structure than that usually used in these approaches. The cells induced by the grid will represent the area in which people are looking for near users or services. Many existing proximity services such as Tinder, allow the user to choose the size of the searching area between some meters to some kilometers (with a given granularity). If we only used a fixed grid, whatever its shape and organization, we could not get flexibility in the size of the searching area. To achieve this flexibility, we design a hierarchical spatial index based on the concept of *quad tree* [44], in which also overlapping is enabled. We call this structure *shifted quad tree* (SQT). A quad tree is a tree in which each internal node has exactly four children. It can be used to partition a 2-dimensional area into regions of different sizes. Specifically, the entire area is associated with the root of the tree and it is partitioned into four regions, each associated with a child of the root. Recursively, each region is partitioned into four regions and so on. The last obtained regions are associated with the leaves of the tree. Figure 5.1 describes the overlapping mechanism of the square cells, obtained by taking two square grids (suppose, one black and the other red) initially coincident and by shifting the red one across the left-bottom diagonal for half diagonal of the square. This way, each user belongs exactly to two squares (one black square and one red square), and two users at a distance less than half of the length of the side of the square have at least one cell in common. This mechanism is implemented at each level of the SQT. To better understand the overlapping mechanism, we always consider the figure 3.1, in which Alice belongs to the black cell $(4,3)$ and the red cell $(3,3)$, Bob belongs to the black cell $(3,2)$ and the red cell $(3,3)$. They both belong to the red cell $(3,3)$, since they are at distance less than half of the length of the side of the square. In the figure, the coordinates of the cells are replicated with different colors since the red grid is thought as shifted from the position of the black grid. So,

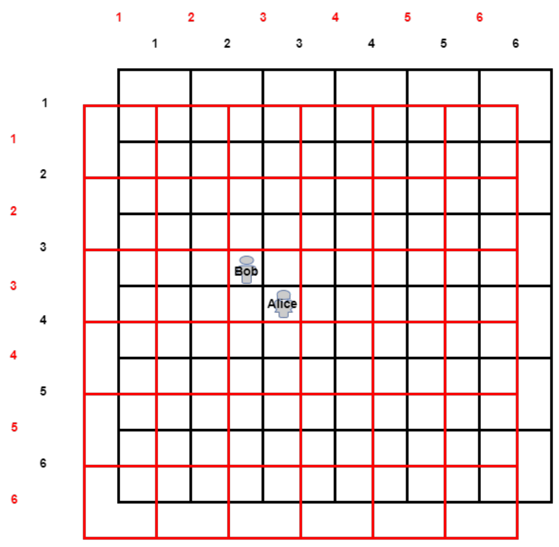


Fig. 3.1: Overlapping mechanism.

.

the red cell (x,y) is the *shifted cell* associated with the black cell (x,y). At each level of a quad tree, we want to keep the same mechanism hierarchically thus obtaining an SQT, which basically is a standard quad tree that includes a *shifted node* for each quad-tree node. The shifted node s of a quad-tree node n indexes, in the territory, a square that is the shifted cell of the square indexed by n. This mechanism is resumed in Figure 3.3, in which we consider a restricted interest area (thus cutting some cells) and a three-level SQT. About coordinates, we need to add two dimensions, one representing the level in the tree and the other needed to make explicit the fact that the coordinates are referring to a quad tree node or to a shifted node. We assume that the numbering of the coordinates starts from 1 for a square of type quad tree entirely included in the interested area.

The coordinates have the following form: $\langle k,i,j,t\rangle$, where k indicates the level (0,1,2, ...), i,j the position, and t the type between *(q)uad* and *(s)hifted*. For example, the cells $\langle 0,1,3,q\rangle$, $\langle 0,1,4,q\rangle$, $\langle 0,2,3,q\rangle$, and $\langle 0,2,4,q\rangle$ (representing four 0-level quad-tree nodes) are aggregated into one 1-level quad-tree node, with coordinates $\langle 1,1,2,q\rangle$. This node indexes the blue square with the same coordinates. The shifted 1-level node associated with this node has coordinates $\langle 1,1,2,s\rangle$ and indexes the green cell with the same coordinates. Moreover, the cells $\langle 1,1,2,q\rangle$, $\langle 1,1,3,q\rangle$, $\langle 1,2,2,q\rangle$, and $\langle 1,2,3,q\rangle$ (representing four 1-level quad-tree nodes) are aggregated into the 2-level quad-tree node coloured in pink. The shifted 2-level node associated with this node is colored in orange.

If we suppose to set a given level, for example k, then we will have that a cell of level k is identified by its center, called *centroid*. Each user is able to identify the

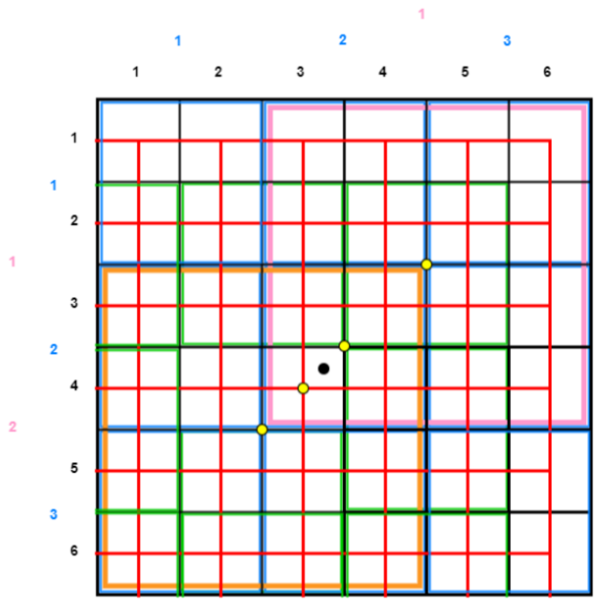


Fig. 3.2: An example of centroids.

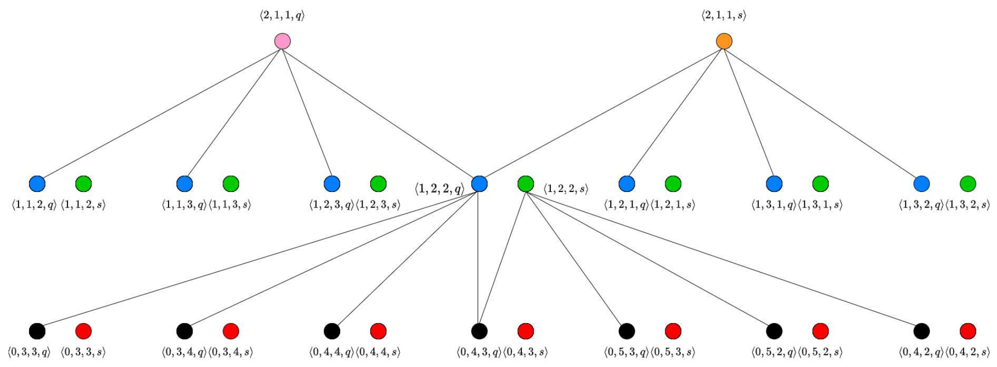

$\langle 2,1,1,q\rangle$
$\langle 2,1,1,s\rangle$
$\langle 1,1,2,q\rangle$ $\langle 1,1,2,s\rangle$
$\langle 1,1,3,q\rangle$ $\langle 1,1,3,s\rangle$
$\langle 1,2,3,q\rangle$ $\langle 1,2,3,s\rangle$
$\langle 1,2,2,q\rangle$
$\langle 1,2,2,s\rangle$
$\langle 1,2,1,q\rangle$ $\langle 1,2,1,s\rangle$
$\langle 1,3,1,q\rangle$ $\langle 1,3,1,s\rangle$
$\langle 1,3,2,q\rangle$ $\langle 1,3,2,s\rangle$
$\langle 0,3,3,q\rangle$ $\langle 0,3,3,s\rangle$
$\langle 0,3,4,q\rangle$ $\langle 0,3,4,s\rangle$
$\langle 0,4,4,q\rangle$ $\langle 0,4,4,s\rangle$
$\langle 0,4,3,q\rangle$ $\langle 0,4,3,s\rangle$
$\langle 0,5,3,q\rangle$ $\langle 0,5,3,s\rangle$
$\langle 0,5,2,q\rangle$ $\langle 0,5,2,s\rangle$
$\langle 0,4,2,q\rangle$ $\langle 0,4,2,s\rangle$

Fig. 3.3: An example of SQT.

centroids of the two cells of level k in which they are located through the use of a localization system. Each centroid can be identified by the coordinates of the cell and, even though a given point in the space could be the centroid of different cells (belonging to different levels), the presence of the coordinates of the level allows us to uniquely identify the centroid. For each level k, a user, being located in two overlapping cells, one of type q and the other of type s, detects two centroids associated with the level k. This operating principle is shown in Figure 3.2, in which the user represented with a black dot, detects, level by level, the centroids $\langle 0,4,3,q\rangle$, and $\langle 0,3,4,s\rangle$, for level 0, the centroids $\langle 1,2,2,q\rangle$, and $\langle 1,2,2,s\rangle$, for level 1, and the centroids $\langle 2,1,1,q\rangle$, and $\langle 2,1,1,s\rangle$ for level 2. The centroids are represented in the figure with yellow dots. Sometimes, a single dot represents multiple coincident centroids (in the territory). Specifically, the centroid $\langle 0,3,4,s\rangle$ coincides with the centroid $\langle 1,2,2,q\rangle$ and the centroid $\langle 1,2,2,s\rangle$ coincides with the centroid $\langle 2,1,1,s\rangle$.

Usually, the proximity services are requested by the users within a maximum radius r with respect to their position. This means that if a service can be provided only with a distance greater than r, it can be excluded. For this reason, the user needs to identify just the cells for each level until a maximum level l (possibly l can coincide with the depth of the SQT). Therefore, we suppose to consider a user X requiring a service inside a maximum radius corresponding to the level l. We denote by C_X^l the set of centroids detected by X at each level less or equal to l. For example, still referring to Figure 3.3, if the black dot represents the user X, then C_X^2 is exactly the set of centroids listed above, i.e., $\langle 0,4,3,q\rangle$, $\langle 0,3,4,s\rangle$, $\langle 1,2,2,q\rangle$, $\langle 1,2,2,s\rangle$, $\langle 2,1,1,q\rangle$, and $\langle 2,1,1,s\rangle$. This information is the basis of the proximity testing implementation, because, in principle, the provider can detect two users in proximity within a certain distance d (among the granularity set induced by the SQT), if they detect the same centroid of a level corresponding to squares of side not greater than $2d$. In Chapter 4, we will see how to exploit this basic principle to obtain a privacy-preserving result, specifically in the service `KN-service`, relating to the proximity of users who know each other.

Our approach is grid-based and so far we described how our grid is hierarchically organized. However, it is also tag-based. We define next how the tag-based mechanism is implemented. In principle, as a *tag*, one of the technologies identified in the literature [120], such as Bluetooth IDs, Wifi IDs, military codes in GPS, audio signals, LTE, and atmospheric gases could be used to obtain an unpredictable value, associated with a point in space and time. The purpose of this value is to allow the users to obscure the centroids when sending them to the provider, by preventing that the provider can reverse the information and then discover the actual positions. This value is called *salt*. A concrete way to implement salts is to rely on the collaboration of a telephone service provider (TSP), which transmits the salts through the cellular network. We can use the cellular cells to identify a region of the space in which, for a given time interval, a random salt, with a suitable rounding protocol, is periodically broadcasted to all the devices belonging to this cell. We can organize the tag-mechanism in a hierarchical fashion (in which the level 0 is represented by the TSP cells), needed to properly combine it with the hierarchical grid, despite the fact that the maximum space granularity we can obtain is the size of the TSP cells. To implement a virtual TSP-cell of level k over the level $k-1$ (for a given $k \geq 1$), it suffices to virtually aggregate a suitable number of virtual TSP cells of level $k-1$ (observe that virtual TSP cells of level 0 are actually physical TSP cells). Virtual/-physical TSP cells are called *tag-cells*. A necessary aspect to understand is how the aggregation of the tag cells is performed to obtain tag cells higher in the hierarchy but first it is worth dealing with the problem of possible misalignment, at a given

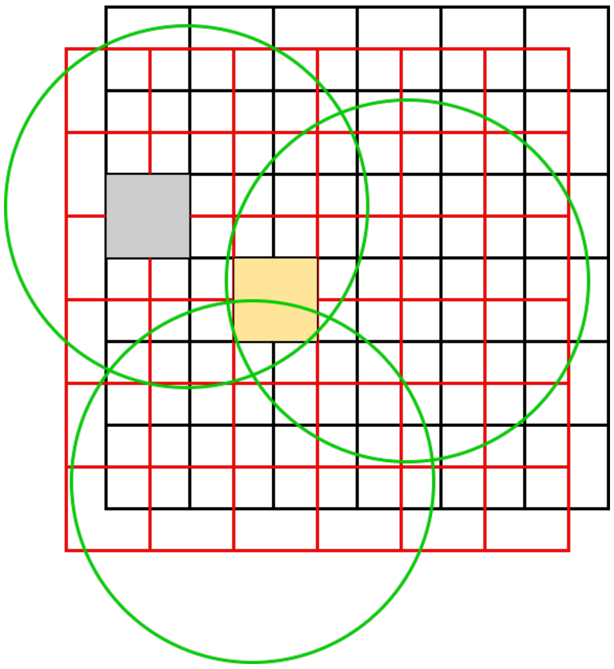


Fig. 3.4: TSP overlapping mechanism.

level of the hierarchy, between the grid and the tag-cell structure. It can happen that a cell of the grid is cut by a tag-cell and, in this case, two users belonging to the same cell might receive two different salts, and then they are not detected in proximity. To avoid this, we enable an overlapping mechanism, which, at level 0 just relies on the physical overlapping between TSP cells adopted to manage the handover, and, at higher levels, it is suitably implemented.

First, we consider the grids of level 0. At level 0, the only requirement is that our cells are of size smaller than the overlapping area of TSP cells. This is realistic because this overlapping can be of some meters. This ensures that two users sharing the same cell (of level 0) receive at least one salt in common.

If we observe Figure 3.4, we can note that the green circles represent the tag-cells of level 0 and all the users in the grey cell receive the same salt, while the users in the yellow cell may receive different salts (according to their positions), of which at least one is in common. This overlapping mechanism has to be achieved at the higher levels, thus we get that some tag-cells of level 0 are aggregated into a tag-cell of level 1. Inside this tag-cell, a new salt (of level 1) should be transmitted. Specifically, the TSP broadcasts in a tag-cell of level 0 both a salt of level 0 and a salt of level 1. The same salt of level 1 is broadcasted in the adjacent tag-cells of level 0 to form the virtual tag-cell of level 1. If we continue to always observe Figure 3.4, we can see that if the three green tag-cells (of level 0) are aggregated into a tag-cell of level 1, the three antennas transmit the same salt of level 1 (and a different salt of level 0 per tag-cell of level 0). By iteratively applying this approach, the tag-cells of level i are aggregated into tag-cells of level $i+1$.

We assume that each salt is sent along with a label indicating its level so that the users can distinguish them. Given a user X, we denote by R^l_X the set of salts of level less or equal to l detected by X at a given instant. We observe that, at a given instant, given two users X and Y and a level l, if X and Y are at a distance less than half of the length of the side of the cells of level l, then $C^l_X \cap C^l_Y \neq \emptyset$ and $R^l_X \cap R^l_Y \neq \emptyset$ i.e., the users share at least one salt and one centroid of level less or equal to l. This observation is the basis of how proximity is detected. The mechanism just described represents the basis on which the research activity conducted was carried out. Specifically, in Chapter 4, this aspect will be dealt with in detail and three different proximity-based services `KN-service`, `UN-service`, `TN-service` will be described.

3.3 Data Integrity of Map Data

Query integrity for outsourced spatial databases is a well-known problem deeply analyzed in the literature. The typical scenario in which the problem is formulated is the provision of LBS (location-based services) by a *service provider* relying on data that are outsourced by a third party, called *data owner*. To obtain the service, the clients submit their queries to the service provider. The problem is to provide the clients with an appropriate level of assurance that the result given by the service provider as a response to a submitted query is *correct* (with no tampered values), *complete* (with no omission) and *fresh* (the most updated data) [147]. A similar scenario with identical needs occurs when the data owner is also the party that exploits data (possibly also allowing external clients). Therefore, it collects and updates data through sensors spread over the territory and outsources them to a third party, typically a cloud. Outsourcing database management to a computationally powerful party makes more feasible intensive and complex operations on data and solves the problem of storing and managing high data volumes. We are referring to the case of huge fine-grained map data, collected by a given entity (possibly a government entity) that relies on a cloud provider to store data and process queries needed to reach the aimed mission. In this case, the data owner itself plays the role of client and needs adequate guarantees that the queries performed by the cloud return correct, complete, and fresh results.

There are a lot of emerging application domains leading to this scenario, such as homeland security, vehicular networks (also by referring to security incident response in the context of autonomous driving), weather maps, and so on. Often, these real-life domains, unlike the case of points of interest (POIs) in LBS, result in dense maps, with highly dynamic data for which freshness is a critical property to fulfill.

The general (proof-based) approach to facing this problem is to add extra authentication information allowing the cloud to return the answer to a query together with a verification object (VO) giving the proof of query integrity.

According to [77, 76, 191], the most significant services in this context may be proximity services, tracking services, and services to locate (nearby) friends or nearby people with similar interests. In this case, the proximity service provider outsources the map data to a third party (typically the cloud), which however may not be honest. The proximity service provider needs guarantees on the completeness and correctness of the portion of map data returned. The question motivating this chapter is to understand if the state-of-the-art techniques used in the context of LBS with outsourced databases can be applied also in our different scenario, or we need to investigate new solutions. The answer is not trivial. Indeed, we have the simplification that we do not need public verifiability of query integrity because the party receiving the response to the query is only the data owner itself. Against, we have a number of challenging issues that complicate the problem: (1) huge and dense map data; (2) very frequent insert and delete operations; (3) need of an effective verification of freshness; (4) need to perform operations changing extra authentication information (for updates) directly at the source, and then through possibly constrained devices (eg., sensors). Concerning the latter, we highlight that to avoid bottlenecks at the data owner back-end, the devices distributed over the territory should perform the updates also by computing updated extra authentication information and by sending them to the cloud. Therefore, the efficiency of updates becomes crucial.

Deterministic approaches existing in the literature basically are of two types (possibly combined): tree-like based or signature-chain based. Tree-like approaches rely on the extension of the notion of Merkle-Hash-Tree [113] to the 2-dimensional case. [180] is probably the most representative proposal, among those that define Merkle-Hash-Tree-based authenticated data structures (ADS) to deal with the case of query integrity for spatial data.

It is well-known that tree-like methods are inherently static, thus not suitable for dynamic contexts [185]. Methods based on signatures, [132, 79, 187, 53] (or other cryptographic mechanisms like accumulation values [133, 186]), suffer from the problem that the construction of the signature-based proofs is more expensive than the tree-based approaches [185].

The idea behind this section was proposed in [42]. The goal is to take advantage from the simplification concerning the non-necessity of public verifiability to adopt a message-authentication-code-based flat method to fulfill the requirements above.

The method allows $O(1)$ insertion and deletion instead of $O(log\ n)$ of tree-like approaches (where n measures the amount of stored data) and does not incorporate the cryptographic complexity of alternate approaches existing in the literature. Moreover, freshness is provided without having to resort to revocation mechanisms and to make assumptions on synchronizations, as happens in the case of public verifiability.

3.3.1 Related Work

Data outsourcing means that data is no longer under the direct control of users and this entails new risks and introduces new challenges for security and privacy. To face the problem of query integrity, a number of probabilistic approaches exist [185]. However, our technique is deterministic. Therefore, in this section, we focus only on deterministic methods. The main approaches are based on trees and signature chains. The tree-based approach uses the Merkle hash tree or its variants. The signature-based approach uses the signature aggregation technique and can lead to low communication complexity, but may require special treatment for handling more powerful queries and often leads to large storage and computation complexities [187].

In [173], and subsequently in [174], Wang et al. propose to combine BLS-based homomorphic linear authenticator with Merkle Hash Tree to support both public auditability and full data dynamics. In [102], Liu et al. present a multi-replica dynamic public auditing scheme (MuR-DPA) which is based on a new multi-replica Merkle hash tree (MR-MHT). With the growing popularity of location-based services and the use of GPS-enabled smartphones and devices, the need to outsource spatial data has grown rapidly in recent years. In [180], the authors propose the MR-tree, an authenticated index based on the Merkle Hash tree and the R*-tree. Yang et al. develop the MR*-tree, an alternative to the MR-tree, which significantly reduces the communication overhead between the LBS and the client; in addition, both the MR-tree and the MR*-tree are fully outsourced. In [106], an authentication data structure based on Merkle hash trees is adopted as an efficient authentication mechanism for checking the integrity of outsourced spatial databases. A new scheme is suggested based on the concepts of Merkle Hash Tree and quad-tree for verifying the authenticity of outsourced spatial databases.

Cheng et al. [53] combined signature chain with a data partitioning structure, considering two schemes, i.e. Verifiable KD-tree (VKDtree) that is based on space partitioning, and Verifiable R-tree (VRtree) that is based on data partitioning. The mechanism must verify that all partitions relevant to the query are returned and that all qualifying data points within each relevant partition are returned. The signature chain technique is used to concatenate points and partitions so that any malicious omissions can be detected by the user. Zheng et al. [187] combine the Homomorphic Linear Authentication (HLA) aggregation signature with MB-tree, by associating the MB-tree entries with the HLA aggregation signatures.

Hu et al. [79] consider the Outsourced Spatial Database (OSDB) model and propose an efficient scheme, called VN-Auth, that allows a client to verify the correctness and completeness of the result set. The approach can handle both k-nearest neighbour (kNN) and range queries, and is based on neighbourhood information derived by the Voronoi diagram of the underlying spatial dataset. VN-Auth produces significantly smaller verification objects (VO) and the client can verify its integrity by examining the signatures and exploring the neighbourhood of every object in the result set. However, insert and delete are not supported and, very dense data would lead to a proliferation of Voronoi regions.

3.3.2 The Basic Data Structure

We start by giving some preliminary definitions. We denote by $H(x)$, the application of a secure cryptography hash function (e.g., SHA256) on a message x and by HMAC(k,M), the application of the HMAC function [95] (with any underlying secure cryptographic hash function) with secret key k on a message M. For both H and HMAC, the output is generically called *digest*.

Given an interval $[a,b] \in \mathbb{R}$, a *marker partition* of $[a,b]$ is a sequence $Z = \langle 0_Z, [z_0, z_1], 1_Z, (z_1, z_2], 2_Z, \ldots, (n\text{-}1)_Z, (z_{n-1}, z_n], n_Z \rangle$ where $z_i \in [a,b]$, $i_Z \notin [a,b]$, for $0 \le i \le n$, $z_0 = a$, $z_n = b$ and $z_i < z_j$ for $0 \le i < j \le n$. The elements i_Z separating the sub-intervals of the sequence are *fresh values*, i.e., they are outside of the domain $[a,b]$. These elements are called *l(inear)-markers (of Z)*.

Given an element α, we say that $\alpha \in Z$ if α is a l-marker or $\alpha \in [a,b]$. In words, the elements of Z are the l-markers and the elements belonging to the intervals in which $[a,b]$ is partitioned. Now, we define a total order among the elements in Z, such that an element $\alpha \in Z$ is less than $\beta \in Z$ if α precedes β in the sequence.

Formally: Given $\alpha, \beta \in Z$ such that $\alpha \neq \beta$, we say that $\alpha <_Z \beta$ if either:
(1) $\alpha = 0_Z$, or
(2) $\alpha, \beta \in [a,b]$ and $\alpha < \beta$, or
(3) $\alpha = i_Z$ and $\beta = j_Z$ are l-markers and $i < j$, or
(4) $\alpha \in [a,b]$, $\beta = i_Z$ (with $i \neq 0$) is a l-marker and $\alpha \le z_i$, or
(5) $\alpha = i_Z$ is a l-marker, $\beta \in [a,b]$ and $\beta > z_i$.

Example 3.1. With this example, we clarify the notions of marker partition, membership, and total order. Consider the interval $[a,b] = [0,100]$ and the marker partition

$Z = \langle 0_Z, [z_0, z_1], 1_Z, (z_1, z_2], 2_Z,$
$(z_2, z_3], 3_Z, (z_3, z_4], 4_Z \rangle$ where $z_1 = 25, z_2 = 50, z_3 = 75$. By definition, $z_0 = 0, z_4 = 100$. Observe that $3_Z \in Z$ because 3_Z is an l-marker. Similarly, $37 \in Z$ because 37 is a value occurring in the interval $[a,b]$ (specifically, $37 \in (z_1, z_2]$). Concerning the total order, it holds that: (1) 0_Z is the minimum, (2) $37 <_Z 78$ as $37 < 78$, (3) $1_Z <_Z 3_Z$ as $1 < 3$, (4) $67 <_Z 3_Z$ as $67 < z_3 = 75$, and (5) $2_Z < 98$ as $98 > z_2 = 50$.

Given an interval $[a,b] \in \mathbb{R}$, an $[a,b]$-*map database* (*of arity* n) is a total function $M : [a,b] \times [a,b] \to A_1 \times \cdots \times A_n \cup \{0_T\}$, where A_is are called *data attributes* and 0_T is the *the zero tuple*, i.e., a dummy tuple which represents the fact that no attribute is associated with a pair (x,y) such that $M(x,y) = 0_T$.

An XY-*markered version* $\bar{M}$ of an $[a,b]$-*map database* M, is obtained by replacing the domain $[a,b] \times [a,b]$ by $X \times Y$, where X and Y are marker partitions of $[a,b]$, in such a way that $\bar{M}(x,y) = M(x,y)$ if $x,y \in [a,b]$, $\bar{M}(x,y) = 0_T$ otherwise. We recall that a marker partition, as defined earlier, is a partition of $[a,b]$ into intervals interleaved by l-markers. $\bar{M}$ is such that the map database is preserved over the pairs belonging to $[a,b] \times [a,b]$ and assigns the dummy tuple 0_T to any other pair with at least an l-marker as a component.

We define a total order on $X \times Y$ such that $(x_i, y_i) <_{XY} (x_j, y_j)$ if $y_i <_Y y_j \vee (y_i = y_j \wedge x_i <_X x_j)$.

Observe that $<_X$ and $<_Y$ are the total orders defined on the marker partitions X and Y, respectively.

We denote by M_X (M_Y, respectively) the (ordered) set of all the l-markers of X (of Y, respectively). The *bucket schema* of $\bar{M}$ is the set $M_X \times M_Y$. An element of $m \in M_X \times M_Y$ is called *marker*.

Given a marker $m = (i_X, j_Y)$ with $i < |M_X| - 1$ and $j < |M_Y| - 1$, we define (i_X, j_Y)-bucket the set $B = \{(x,y) \in [a,b] \times [a,b] : i_X <_X x <_X (i+1)_X, j_Y <_Y y <_Y (j+1)_Y\}$.

Example 3.2. In Fig. 3.5, the domain $X \times Y$ of a XY-markered version $\bar{M}$ of an $[a,b]$–map database, is depicted. The X and Y markers partitions of $[a,b]$ are obtained by dividing $[a,b]$ in 4 sub-intervals of equal size. The red elements are the markers. The blue elements represent pairs (x,y) such that $\bar{M}(x,y) \neq 0_T$, i.e., they are elements with some associated information. For all other elements (x,y) not represented in the figure, we assume $\bar{M}(x,y) = 0_T$. Finally, the green square represents the $(1_X, 1_Y)$-bucket.

Given an (i_X, j_Y)-bucket B, and $t \geq 0$, we define a *t-ghost chain* on B, the (possibly empty) sequence $G_B = \langle g_1, g_2, \ldots, g_t \rangle$ such that:
(1) For each $1 \leq q < t$, $g_q = (d_q, e_q)$, where d_q is a digest and $e_q =$HMAC$(k, d_q \| d_{q+1})$,
(2) $g_t = (d_t, e_t)$, where d_t is a digest and $e_t =$HMAC$(k, d_t \| H(((i+1)_X, (j+1)_Y) \| 0_T))$.

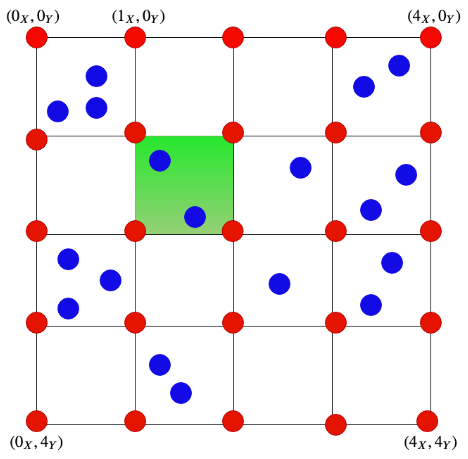


Fig. 3.5: Domain of a XY-markered version $\bar{M}$ of an $[a,b]$-map database.

An element $g \in G_B$ is called *ghost tuple (g-tuple)*.

Given an XY-markered version $\bar{M}$ of an $[a,b]$-map database M, an (i_X, j_Y)-bucket B, and a t-ghost chain G_B, we define the k-integrity chain on B, as the sequence $V = (v_1, v_2, .., v_k)$ of 3-tuples $v_q = (a_q, b_q, c_q)$, such that:

(1) $a_1 = (i_X, j_Y)$, $a_k = ((i+1)_X, (j+1)_Y)$,

(2) for $1 < q < k$, $a_q \in B$ and $\bar{M}(a_q) \neq 0_T$,

(3) for $1 \leq q \leq k$, $b_q = \bar{M}(a_q)$,

(4) for $1 \leq q < k$, if $1 \leq q < k-1 \vee (q = k-1 \wedge G = \emptyset)$, then $c_q =$HMAC$(k, H(a_q \| b_q) \| H(a_{q+1} \| b_{q+1}))$, else, $c_{k-1} =$HMAC$(k, H(a_{k-1} \| b_{k-1}) \| d_1)$,

(5) c_k is any digest (we do not care the value),

(6) $V = (v_1, v_2, .., v_k)$ is ordered on a_q according to the relation $<_{XY}$.

The elements $v_1, v_k \in V$ are called *marker-tuples (m-tuples)* and the elements $v_q \in V$ with $1 < q < k$ are called *secured-tuples (s-tuples)*.

A set $R = [a', b'] \times [c', d'] \subseteq [a,b] \times [a,b]$ is called *range set*. Given a range set R and a k-integrity chain V on a bucket B, a s-tuple $v_q = (a_q, b_q, c_q) \in V$ is called *in-tuple* (for R) if $a_q \in R$, otherwise v_q is called *out-tuple* (for R). A bucket B composed only of in-tuples for R (out-tuples for R) is called *internal* bucket for R (*external* bucket for R, resp.). Any other bucket is called *borderline* bucket for R.

Given a range set R and an k-integrity chain V on a bucket B, we define the R-modified version $\bar{V}$ of V as the sequence $\bar{V} = (\bar{v}_1 \ldots, \bar{v}_k)$ of 3-tuples $\bar{v}_q = (\bar{a}_q, \bar{b}_q, \bar{c}_q)$, such that:

(1) $\bar{v}_1 = v_1$ and $\bar{v}_k = v_k$,

(2) for $1 < q \leq k-1$, if $v_q = (a_q, b_q, c_q)$ is an out-tuple, $\bar{v}_q = (a_q, H(a_q \| b_q), c_q)$ (thus obtaining a *succinct version* of the out-tuple), $\bar{v}_q = v_q$ otherwise.

Observe that, as discussed in Section 3.3.4, the succinct version of an out-tuple consists of a few bits that do not significantly make worse the size of the VO.

A *secured version* M_s of an $[a,b]$-map database M consists of:
(1) An XY-markered version $\bar{M}$ of M,
(2) For each bucket B, an integrity chain V on B (with ghost chain G_B),
(3) For each integrity chain V on B, a state S_B which counts the number of insert and delete operations affecting B.

On a secured version M_s of an $[a,b]$–map database, the following operations are allowed:
INSERT: Given a bucket B, this operation receives, as input, an element $a \in B$ such that $\bar{M}(a) = 0_T$ and an element $b \in A_1 \times \ldots \times A_n$. It updates the function $\bar{M}$ in such a way that $\bar{M}(a) = b$. Then, it creates a new 3-tuple v and inserts it in the integrity chain V on B. Finally, it increments by one the state S_B.
DELETE: This operation receives, as input, an element $a \in B$. It updates the function $\bar{M}$ in such a way that $\bar{M}(a) = 0_T$. Then, it removes the 3-tuple v (if any), with a as left-most component, from the integrity chain V on B and adds a new g-tuple g to the end of the sequence G_B. Finally, it increments by one the state S_B.
RANGE_QUERY: This operation receives, as input, a range query $Q(R)$ where R is a range set. It outputs all the elements in R along with additional information (VO) to verify the integrity of the result.
VERIFY: This operation receives, as input, the output of the RANGE_QUERY operation and outputs **true** if the query integrity is checked (in the sense defined in the next section), **false** otherwise.

3.3.3 Query Integrity Protocol

In this section, we describe how, on the basis of the data structure introduced in the previous section, it is possible to define a protocol to guarantee query integrity on map data. Query integrity is guaranteed if three properties are fulfilled: *Completeness* (all the tuples involved in the queries have to be returned), *Correctness* (the tuples returned have not to be corrupted), and *Freshness* (the most up-to-date version of tuples has to be returned). The actors of our protocol are:
Data Owner. It is the party that owns map data and outsources these to the cloud. It consists of two entities: the back-end and the sensors. The back-end submits the range queries to the cloud and verifies their integrity. The sensors insert data (with high frequency) and must secure data at the source, to avoid possible compromising. We assume that sensors are constrained devices, so that our data structures are designed in such a way that insert and delete operations are very efficient. The back-end keeps and updates, for each bucket, the state information (i.e., an integer value). Finally, the key k is shared between the sensors and the back-end.

Cloud. It stores data and processes range queries. The cloud keeps, for each bucket, the associated integrity chain.

The bucket schema $M_X \times M_Y$ is public and, thus not tamperable. We recall that the elements of an integrity chain V are 3-tuples of the form $v_q = (a_q, b_q, c_q) \in V$, the elements of a ghost chain G_B are pairs of the form $g_q = (d_q, e_q) \in G_B$ and the elements of a modified version $\bar{V}$ of an integrity chain V are 3-tuples of the form $\bar{v}_q = (\bar{a}_q, \bar{b}_q, \bar{b}_q) \in \bar{V}$.

As introduced in the previous section, the query integrity protocol consists of four operations: INSERT, DELETE, RANGE_QUERY, VERIFY. Since INSERT and DELETE update the state of a bucket, we define an auxiliary operation called STATE_UPDATE which involves the back-end to update the state. The way in which this operation is implemented guarantees the integrity of the state update.

INSERT.

This operation affects both the map database and an integrity chain. Indeed, the integrity chain would include a new element corresponding to the new inserted tuple. The sensor starts by sending to the cloud the coordinates in which the insertion should be done together with the information being inserted. Let B be the involved bucket. The cloud responds by sending to the sensor the portion of the integrity chain (two elements – see below) necessary to perform the insertion of the new element. Indeed, the cloud cannot directly perform this operation because it does not own the secret key k needed to compute HMACs. The sensor contacts the back-end to update the state information associated with the bucket. The cloud receives and includes the updated portion of the integrity chain.

In detail, let $a \in B$ be the insertion point and $b \in A_1 \times \ldots \times A_n$ be the tuple being inserted. Given the k-integrity chain V on B, the g-chain G_B, and the state S_B, this operation performs the steps described in the following:

(1) The sensor sends to the cloud a, b;

(2) the cloud finds the elements $v_q, v_{q+1} \in V$ such that $a_q <_{XY} a <_{XY} a_{q+1}$;

(3) if $q = k-1$ and $G_B \neq \emptyset$, the cloud computes $h_1 = H(d_1)$ and $h_2 = H(a_q \| b_q)$, else it computes $h_1 = H(a_{q+1} \| b_{q+1})$ and $h_2 = H(a_q \| b_q)$;

(4) the cloud sends h_1, h_2 to the sensor;

(5) the sensor computes $h_3 = H(a \| b)$, $h_4 = \text{HMAC}(k, h_3 \| h_1)$, $h_5 = \text{HMAC}(k, h_2 \| h_3)$;

(6) the sensor sends h_4, h_5 to the cloud and invokes STATE_UPDATE;

(7) the cloud creates the elements $v = (a, b, h_4)$ and inserts v between v_q and v_{q+1};

(8) finally, the cloud sets $c_q = h_5$.

Formally, the insert operation changes the value of the function $\bar{M}$ in a from 0_T to b. Observe that the links of the chain so obtained consist of HMACs computed by the sensor. The aim of this solution is to make them computable only by the data owner in such a way that no fake insertion can be simulated by the cloud.

DELETE.

This is an operation dual w.r.t. insertion. Therefore, the integrity chain must be also updated by dropping the element corresponding to the deleted tuple. However, in this case, it is also needed to include a new g-tuple in the g-chain. This allows the data owner to detect missing deletions (as better explained in Section 3.3.6). Let $a \in B$ be the point in which the deletion must be done. Given the k-integrity chain V on B, the t-ghost chain G_B and the state S_B, this operation performs the steps in the following:

(1) The sensor sends a to the cloud;

(2) the cloud finds the elements $v_{q-1}, v_q, v_{q+1} \in V$ such that $a_q = a$;

(3) the cloud computes $h_1 = H(a_{q-1} || b_{q-1})$;

(3) if $q = k-1$ and $G_B \neq \emptyset$, the cloud sets $h_2 = d_1$, else it computes $h_2 = H(a_{q+1} || b_{q+1})$;

(4) if $G_B \neq \emptyset$, the cloud sets $h_3 = d_t$, else if $q = k-1$, it computes $h_3 = H(a_{k-2} || b_{k-2})$, else it computes $h_3 = H(a_{k-1} || b_{k-1})$;

(5) the cloud computes $h_4 = H(a_k || b_k)$;

(6) the cloud sends $h_1, h_2, h_3, h_4, c_{q-1}, c_q, b_q$ to the sensor;

(7) the sensor computes $h_5 = H(a || b_q)$ and checks that $c_{q-1} =$ HMAC$(k, h_1 || h_5)$ and that $c_q =$HMAC$(k, h_5 || h_2)$;

(8) the sensor computes $h_6 =$HMAC$(k, h_1 || h_2)$, $h_7 =$HMAC(k, h_5), $h_8 =$HMAC$(k, h_3 || h_7)$, and $h_9 =$HMAC$(k, h_7 || h_4)$;

(9) the sensor sends (h_6, h_7, h_8, h_9) to the cloud and invokes STATE _UPDATE;

(10) the cloud sets $c_{q-1} = h_6$;

(11) if $G_B \neq \emptyset$, the cloud sets $e_t = h_8$, else if $q = k-1$, it sets $c_{k-2} = h_8$, else it sets $c_{k-1} = h_8$

(12) the cloud creates the g-tuple $g = (h_7, h_9)$ and inserts it at the tail of G_B.

Formally, the delete operation sets the value of the function $\bar{M}$ to 0_T. The check at the step (8) ensures that the cloud returns to the sensor the correct digests. In fact, without this check, the cloud could re-use an old HMAC to link v_{q-1} and v_{q+1} and obtain a new HMAC for an arbitrary element. Note that this check is not necessary for the INSERT operation because the HMAC computations performed by the sensor involve the new tuple to be inserted, thus the cloud could not obtain arbitrary HMACs.

RANGE_QUERY.

Let be $Q(R)$ the range query with Range Set R:

(1) The back-end sends $Q(R)$ to the cloud;

(2) the cloud builds the set $\mathbb{V}$ containing all the pairs (V, G_B), where B is an internal bucket for R and V is the integrity chain on B with ghost chain G_B;
(3) the cloud builds the set $\bar{\mathbb{V}}$ containing all the pairs $(\bar{V}, G_B)$, where B is a borderline bucket for R and $\bar{V}$ is the modified version of the integrity chain V on B with ghost chain G_B.
(4) the cloud sends $\mathbb{V}, \bar{\mathbb{V}}$ to the back-end.

VERIFY.

First to define the VERIFY function, we introduce a modified hash function $\bar{H}$ defined as: $\bar{H}(x\|y) = y$ if y is a digest, $\bar{H}(x\|y) = H(x\|y)$ otherwise.

This operation is performed entirely by the back-end which, after submitting the range query $Q(R)$, receives the sets $\mathbb{V}, \bar{\mathbb{V}}$.

The steps are the following:

1. For each $(V, G_B) \in \mathbb{V}$, where V is the k-integrity chain on the bucket B with t-ghost chain G_B and state S_B:
   a) Check c_q =HMAC$(k, H(a_q\|b_q)\|H(a_{q+1}\|b_{q+1}))(1 \le q < k-1)$
   b) If $G_B \neq \emptyset$,
      i. Build the set $D = \{\text{HMAC}(k, H(a_q\|b_q))\}$ $(2 \le q \le k-1)$
      ii. Check that c_{k-1} =HMAC$(k, H(a_{k-1}\|b_{k-1})\|d_1)$
      iii. For $1 \le q \le t-1$, check that e_q =HMAC$(k, d_q\|d_{q+1})$
      iv. For $1 \le q \le t-1$, check that $d_q \notin D)$
      v. Check that e_t =HMAC$(k, d_t\|H(a_k\|b_k)))$
   c) If $G_B = \emptyset$, check c_{k-1} =HMAC$(k, H(a_{k-1}\| b_{k-1})\|H(a_k\|b_k))$
   d) Check that $S_B = k-2+2t$
2. For each $(\bar{V}, G_B) \in \bar{\mathbb{V}}$, where $\bar{V}$ is the R-modified version of the k-integrity chain V on B with t-ghost chain G_B and state S_B:
   a) Check $\bar{c}_q$ =HMAC$(k, \bar{H}(\bar{a}_q\|\bar{b}_q)\|\bar{H}(\bar{a}_{q+1}\|\bar{b}_{q+1}))$ $(1 \le q < k-1)$
   b) If $G_B \neq \emptyset$,
      i. Build the set $D = \{\text{HMAC}(k, \bar{H}(\bar{a}_q\|\bar{b}_q))\}$ $(2 \le q \le k-1)$
      ii. Check that $\bar{c}_{k-1}$ =HMAC$(k, \bar{H}(\bar{a}_{k-1}\|\bar{b}_{k-1})\|d_1)$
      iii. For $1 \le q \le t-1$, checks that e_q =HMAC$(k, d_q\|d_{q+1})$
      iv. For $1 \le q \le t-1$, check that $d_q \notin D)$
      v. Check that e_t =HMAC$(k, d_t\|\bar{H}(\bar{a}_k\|\bar{b}_k)))$
   c) If $G_B = \emptyset$, check $\bar{c}_{k-1}$ =HMAC$(k, \bar{H}(\bar{a}_{k-1}\|\bar{b}_{k-1})\|\bar{H}(\bar{a}_k\| \bar{b}_k))$
   d) Check that $S_B = k-2+2t$

If all the previous checks are verified, then this operation returns **true**, otherwise returns **false**.

Now, we briefly discuss how the checks performed in this operation ensure query integrity. Anyway, the security analysis is treated in Section 3.3.6.

First, the checks (1.a), (1.b.ii), (1.b.iii), (1.b.v), (1.c) and (2.a), (2.b.ii), (2.b.iii), (2.b.v), (2.c) verify that the elements in the integrity chains and in the ghost chains are correctly linked through the HMACs, so that the cloud can not tamper their values. The checks (1.b.iv) and (2.b.iv) explain the role of the g-tuples. In fact, they ensure the cloud does not omit a deletion or performs an unauthorized deletion in place of a legal one. Finally, the checks (1.d), (2.d) use the notion of state, to guarantee the freshness of the result. We recall that the state of each chain is increased by one when an insertion or a deletion occurred. The term $k-2$ is the number of s-tuples occurring in an integrity chain (one per insertion). t is the number of g-tuples occurring in the corresponding g-chain (one per deletion). Moreover, for each g-tuple, a previous s-tuple has been previously inserted in the chain, thus the term t is multiplied by 2. $S_B \neq k-2+2t$ means that the cloud omitted an operation between INSERT and DELETE and tries to deceive the data owner by returning an old version of the chain.

STATE_UPDATE.

We recall that this operation is invoked by the INSERT and DELETE operations. Consider the (i_X, j_Y)-bucket B in which a tuple has to be inserted or deleted. The steps are the following:

(1) The sensor picks up a random R and computes h =HMAC$(k, R\|i_X\|j_Y)$ and sends $R, (i_X, j_Y), h$ to the back-end.

(2) The back-end computes h'=HMAC$(k, R\|i_X\|j_Y)$ and checks that $h' = h$ and that the random R has never been used before for the bucket B.

(3) The back-end increases S_B by one.

Example 3.3. In Fig. 3.6, we extend the example of the previous section by introducing the integrity chains. The red elements are the markers and the blue elements are the s-tuples. For each bucket, the arrows which link two elements represent the fact that these elements are consecutive in the integrity chain associated with the bucket. The yellow elements are the g-tuples forming the g-chain associated with a bucket. For graphical reasons, we represent the g-tuples inside the bucket, but actually, they are not in the domain $X \times Y$. In this example, the only buckets with non-empty g-chain are $(0_X, 0_Y), (3_X, 0_Y), (2_X, 2_Y)$-buckets. This means that in all other buckets, no deletion occurred. Now, we consider that the data owner submits a range query with Range Set R represented by the black rectangle in the figure. The $(2_X, 2_Y)$-bucket is an internal bucket for R, while the others 8 buckets, inside the green rectangle, are borderline buckets for R. The cloud outputs two sets of integrity chains on the buckets inside the green rectangle. The first set contains only an integrity chain on

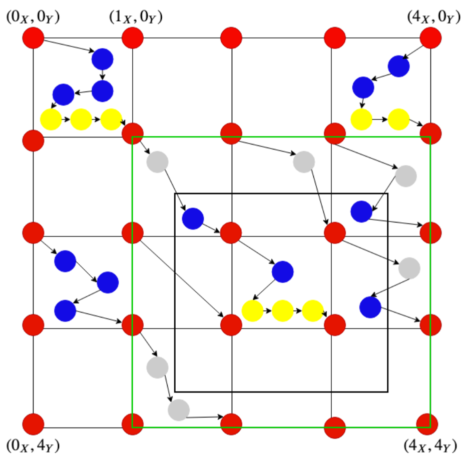


Fig. 3.6: Range query.

the internal $(2_X, 2_Y)$-bucket and the second set contains 8 R-modified versions of integrity chains (one for each borderline bucket). The grey elements are out-tuples for R and are not required directly by the data owner, but they are necessary to verify the integrity of the query. However, in a R-modified version of an integrity chain, when an out-tuple occurs, it is replaced with a more lightweight version which includes the digest of the associated information instead of the information itself.

3.3.4 Cost Analysis

In this section, we provide a cost analysis, in terms of time and space required by our solution. Time-costs are expressed in terms of number of *elementary* hash function applications, which is the application of the function on a message with size not greater than the size of blocks on which it is defined (according to the Merkle-Damgard scheme). For example, SHA-256 works by diving the message into blocks of size 512 bits.

We denote by l the number of bits of the elementary block of a given hash function and by p the number of bits of the digest. For instance, for SHA-256, l =512 bits and $p = 256$ bits. We do not consider the cost of concatenation, assignment, padding, and XOR, because they are negligible compared to the hash computations. To be general, we analyze the cost asymptotically by considering as variable: (1) the number n of tuples of the entire database, (2) the quantity $\frac{x}{l}$, representing the number of elementary blocks (for the hash function) into which a tuple of size x is divided; (3) the number q of tuples required by the query (when applicable).

TIME-COST ANALYSIS.

Now, we analyze the cost of the operations performed by the data owner, which is usually considered in the field of query integrity the most critical issue.

On the basis of the preliminary notations, it is easy to see that the computational cost of the application of H on a message M of size x, is $C(x) = \frac{x}{l}$.

All operations need the computation of the HMAC function. The definition of HMAC is: $\text{HMAC}(K,M) = H((K \oplus opad)||H((K \oplus ipad)||M)$, where K is a secret key of arbitrary size $|K|$ assumed equal to l. It is possible to compute $K \oplus opad$ and $K \oplus ipad$ only once and these values can be stored and used when required. Moreover, we can neglect also the cost of the XOR operation. $((K \oplus ipad)||M)$ returns a block of size $l+x$. Therefore, $H((K \oplus ipad)||M)$ costs $\frac{l+x}{l} = 1 + \frac{x}{l}$. The hash function H operates on $(K \oplus opad)||H((K \oplus ipad)||M)$, which has size $l + p$. We can assume that $p < l$. The message of size $l+p$ is padded to the size $2l$. Consequently, the overall cost is: $\frac{2l}{l} + 1 + \frac{x}{l} = 3 + \frac{x}{l}$.

Now, we can determine the computational costs of the operations INSERT, DELETE, and VERIFY. We omit RANGE_QUERY because it does not involve hash computation for the data owner.

Obviously, we are interested in the data-owner-side computational cost. Observe that, in our model, the data owner consists of two components, that are the sensors and the back-end. We focus our analysis on sensor-side operations, because sensors can be, in general, constrained devices. On the other hand, both INSERT and DELETE invoke STATE_UPDATE which involves also the back-end. Besides the fact that the back-end is not constrained, STATE_UPDATE requires for the data owner just a single hash operation, so it is not relevant for our analysis. Instead, the VERIFY procedure is entirely performed by the back-end and requires several hash computations. Therefore it is included in the computational analysis.

We denote by x the size of $a||b$, where $a \in [a,b] \times [a,b]$ and $b \in A_1 \times \cdots \times A_n \cup \{0_T\}$ (assuming that such a size is the same for each $a||b$). Moreover, we denote by p the size of the digests of H and assume to use a hash function for which $2p = l$ (e.g., SHA256).

INSERT: The hash computations performed by the sensor are h_3, h_4, h_5 (Step (5)) and h in the step (1) of STATE_UPDATE. The computation of $h_3 = H(a||b)$ can be performed with cost $\frac{x}{l}$. Instead, h_4, h_5 are computed by applying the HMAC function on the concatenation between two digests ($h_3||h_1$ for h_4 and $h_2||h_3$ for h_5, respectively). Thus, the input of the HMAC function has size $l = p+p$ and the cost of each HMAC is $3 + \frac{l}{l} = 4$. Regarding h, we assume that the size of $R||i_X||i_Y$ is less than l, then because the padding, the input of HMAC has size l and the cost is 4. The total cost is then: $C_{INS} = \frac{x}{l} + 2 \cdot 4 + 4 = \frac{x}{l} + 12$. Therefore, C_{INS} is $O(\frac{x}{l})$.

DELETE: The reasoning is similar to the case of INSERT. The hash computations performed by the sensor are $h_5, c_{q-1}, c_q, h_6, h_7, h_8,$
h_9 (Step (8-9)) and h in the STATE_UPDATE operation. The computation of $h_5 = H(a||b_q)$ can be performed with cost $\frac{x}{l}$. Regarding the digests $c_{q-1}, c_q, h_6, , h_8, h_9$, they

are computed by applying the HMAC function on the concatenation between two digests. Thus, the input of the HMAC function has size $l = p + p$ and the cost of each HMAC is $3 + \frac{l}{l} = 4$. Finally, the digest h_7 is obtained by applying, again, the HMAC function with an input of size p bits, but it is padded to the size of l bits, so, again, the cost of HMAC is 4. Clearly, the cost of the computation of h is, again, 4. The total cost is then: $C_{DEL} = \frac{x}{l} + 6 \cdot 4 + 4 = \frac{x}{l} + 28$. Therefore, C_{DEL} is $O(\frac{x}{l})$

VERIFY: For this operation, the evaluation of the computational costs is a bit more complicated than the previous. In fact, it depends on the number of (internal and borderline) buckets involved in the query and, for each of them, the cost depends on the number of in-tuples, ex-tuples, and g-tuples. We denote by $N(i)$, $N(o)$, $N(g)$, $N(m)$ the total number of in-tuples, out-tuples, g-tuples, and m-tuples, respectively, involved in a range query. We highlight that $N(i)$ are the actual tuples that the data owner requires in the range query, therefore $N(i) = q$. The others $N(o)$+$N(g)$+$N(m)$ elements are additional information used to verify query integrity.

For the sake of brevity, we omit some details and provide directly the total cost of this operation:
$C_{VERIFY} = (q + N(m)) \cdot \frac{x}{l} + 4 \cdot (q + N(m) + N(g) + N(o)) + 4 \cdot (q + N(o))$.

We assume that $N(o)$ and $N(g)$ are $\Theta(q)$, and that $N(m)$ is $O(q)$. Therefore, C_{VERIFY} is $O\left(q \cdot \frac{x}{l}\right)$. Observe that, as expected, none of the INSERT, DELETE and VERIFY operation depends on the total number of tuples of the database n.

We remark that in the proposed scenario, the operation VERIFY is performed by the back-end, by using back-office resources, thus not constrained. For this reason, we do not consider this aspect critical in our analysis.

SPACE-COST ANALYSIS.

Another important metric to consider is the overhead, in terms of (exchanged and stored) bits that our proposal requires. In particular, we consider:
(i) the cloud-side additional storage (as the data owner does not maintain any data structure), and
(ii) the size of the verification object (VO).

We extend the notation of Section 3.3.4, specifically with reference to time analysis, by introducing $N_t(m)$ and $N_t(g)$ that denote the total number of markers and g-tuples in the database, respectively.

First, consider item (i). The bucket schema is public and intentionally generable so that it should not be materialized and stored by the cloud. However, for each marker the cloud has to keep one digest. In addition, the cloud maintains one digest per tuple. Finally, as overhead, the cloud keeps the set of g-tuples, each consisting of two digests. In sum, we have the amount of bits equal to: $p \cdot (n + N_t(m)) + 2 \cdot p \cdot N_t(g)$, where, we recall, p is the number of bits of digests.

We assume that $N_t(m)$ and $N_t(g)$ are $O(n)$. Therefore, the cloud-side additional storage is $O(n)$.

Regarding (ii), the VO is represented by the in-tuples, the succinct version of involved out-tuples (if any), the g-tuples (if any), and the m-tuples (that are, we recall, the head and the tail markers of the chains associated with the returned buckets).

The VO Size is then (in bits): $(x+p)\cdot q + o \cdot N(o) + 2 \cdot p \cdot N(g) + e \cdot N(m)$, where o and e denote the size, in bits, of a succinct version of an out-tuple and of a m-tuple, respectively.

Observe that both o and e are (constant) small values (of the order of 2-3 times p). Moreover, we assume that $N(o)$,$N(g)$, and $N(m)$ are $O(q)$. Therefore, the VO size is $O(q \cdot x)$

3.3.5 Performance Comparison

In this section, we provide a first preliminary validation of our approach by comparing it with MHT-based techniques (MHT stands for Merkle Hash Tree). We start by performing an analytical comparison. The computational cost of our solution in terms of time and space has been discussed in 3.3.4. Regarding MHT-based approaches, there are different variants of Merkle Hash Trees.

Observe that we can easily determine lower-bound costs by considering that insertion and deletion require in any case the computation of the root, so they include a logarithmic contribution on the number of leaves of the tree (i.e., n). Moreover, insertion requires also the computation of the hash of the new tuple (i.e., $\frac{x}{l}$). Finally, for the verification, for any tuple, at least the hash must be computed and the path until the root must be recomputed (i.e., $q \cdot \frac{x}{l} + log\ n$).

By considering the results described in Section 3.3.4 (concerning our technique relating to time analysis), we obtain the Table reported in Table 3.1. Note that the performance of our solution does not depend on the total number of the elements in the database, and this makes our approach very scalable. We observe that the asymptotic improvement w.r.t. MHT-based approaches for INSERT and VERIFY is at least the reduction from $log\ n$ to a constant cost. Even though $log\ n$ is not a cost treating scalability, we can say that, for big data, the improvement can give real benefits, also by observing that for MHT-based approaches we only consider lower-bound costs valid for each technique of that type. For DELETE, the reasoning is the same, provided that $\frac{x}{l}$ is considered constant. Obviously, it does not depend on n. However, an analysis on the size of tuples is not meaningless. This would lead to the conclusion that for very large tuples (this is not the case of sensors, for example), concerning only DELETE, the benefits of our technique are reduced.

Operation	Tree-based	Our proposal
INSERT	$\Omega(\frac{x}{l} + log\ n)$	$O(\frac{x}{l})$
DELETE	$\Omega(log\ n)$	$O(\frac{x}{l})$
VERIFY	$\Omega(q \cdot \frac{x}{l} + log\ n)$	$O(q \cdot \frac{x}{l})$

Table 3.1: Time-Cost comparison.

Type of analysis	Tree-based	Our proposal
Cloud-side Additional Storage	$\Omega(n)$	$O(n)$
VO Size	$\Omega(q \cdot x + log\ n)$	$O(q \cdot x)$

Table 3.2: Space-Cost comparison.

Now, we consider the space-cost analysis. Regarding the cloud-side additional storage, for MHT-based approaches, at least a digest per tuple has to be maintained by the cloud. Thus, the total required space is $\Omega(n)$.

Regarding the size of the VO, in MHT-based approaches, except for some few extreme cases, the VO includes, along with the tuples required in the query, at least a digest per node in the path from the leaves of the tree and the root. Then, the VO size is $\Omega(q \cdot x + log(n))$. The result agrees with what is reported in [185].

By considering the results described in Section 3.3.4 (concerning our approach relating to space analysis), we obtain the Table 3.2.

The above analysis highlights that our proposal has some benefits also in terms of space.

3.3.6 Security Analysis

In this section, we conduct a security analysis of our protocol. The considered actors are the cloud and the data owner, which owns the secret key k used to compute the HMAC function. In our adversary model, the attacker is the cloud. The basic *assumptions* we make are:

A1: the bucket schema $M_X \times M_Y$ is public and not alterable.

A2: the data owner maintains the state S_B for each bucket B and it is not alterable by the attacker.

A3: the secret key k of the data owner is not guessable.

A4: the hash function used are unbreakable (pre-image resistant, second pre-image resistant, collision-resistant).

We define the following *security compromises*:

$\mathbf{C_{frs}}$. The query result of a range query is *compromised on freshness* if it differs from the actual query result and is obtained from an old (valid) integrity chain.

$\mathbf{C_{crr}}$. The query result of a range query is *compromised on correctness* if it differs from the actual query result and at least one of the following cases holds: (1) There exists in the query result an i-tuple obtained by the cloud by altering an original tuple, or (2) A new i-tuple is forged by the cloud and included in the query result.

$\mathbf{C_{cmp}}$. Given a tuple t internal to the range query in the original map database, the query result of a range query is *compromised on completeness* if t does not occur in it.

The query result of a range query is *integrity-proven* if $\neg C_{frs} \wedge \neg C_{crr} \wedge \neg C_{cmp}$ holds.

We could show that if the VERIFY operation is satisfied, then the following three invariants hold. Informally, the invariants are: **I1** all the HMAC links are correct; **I2** the states S_Bs are coherent w.r.t. the returned number of tuples; **I3** No returned tuple (out-tuple or in-tuple) correspond to g-tuples. Note that the invariant *I2* is based on the fact that the state S_B is the correct value and cannot be tampered by an attacker. It easy to check that this is true since the state is maintained and updated by the data owner. The update is done when the sensor invokes the STATE_UPDATE operation in which it sends an authenticated information to the back-end which cannot be generated by an attacker.

We show now that the logic conjunction of the three invariants introduced above is sufficient to prove that the query result is integrity-proven, and thus the security of our approach. In other words, this proves the effectiveness of the VERIFY operation as a way for the data owner to have assurance about the integrity of the range-query results returned by the cloud.

Theorem 3.4. The query result of a range query is integrity-proven if it the VERIFY operation returns true.

Proof. The proof consists in showing, by contradiction, that if the query result is not integrity-proven (i.e., at least one of $C_{frs}, C_{crr}, C_{cmp}$ holds) then the VERIFY operation returns false (i.e., at least one of the three invariants does not hold).

First, it is possible to show that if C_{frs} holds, then *I2* does not hold. Indeed, by definition of C_{frs}, the query result is obtained from an old integrity chain. This means that at least one INSERT or DELETE operation is omitted by the cloud. However, the state S_B is anyway incremented by one at each operation by the owner. Therefore, it is not coherent with the number of tuples returned in the query result (see steps 1.d and 2.d of the VERIFY procedure) and then the invariant *I2* is violated.

Consider now C_{crr}. If it holds, then at least one among *I1*, *I2*, *I3* does not hold. This can be proven by showing that if C_{crr} holds, to maintain *I1* valid, either *I2* or *I3* does not hold. Indeed, in the case both C_{crr} and *I1* hold, as HMACs can not be forged from scratch, the only possibility is that they are obtained from the sensors during the execution of the protocol. To do this, there are two possibilities: (1) the HMACs are obtained from a legal execution of the protocol and (2) the HMACs are obtained by deceiving the sensors during the INSERT and DELETE operation. The case (1) occurs when the sensor inserts a tuple and, successively, removes it. The cloud can use the HMACs obtained from the INSERT operation to replace or create a new tuple. However, if the cloud does not alter the g-chain associated with the bucket where the tuple was inserted, a g-tuple corresponding to this tuple exists and the invariant *I3* does not hold. On the other hand, if such a g-tuple is removed from the g-chain, *I2* does not hold. Regarding the case (2), the way in which INSERT and DELETE are implemented avoids this possibility without invaliding *I2* or *I3*. In a similar way, it is possible to show that if C_{cmp} holds, then at least one of *I1*, *I2*, *I3* does not hold. The proof sketch is then concluded.

4

A Privacy-Preserving Protocol for Proximity-Based Services in Social Networks

Several innovative applications could be advantageously placed within social networks, to be effective, attractive, and pervasive. An example of an application domain that could benefit from social networks is proximity-based services. This feature exposes users to serious privacy threats because it could allow for massive monitoring by an honest but curious provider. In the literature, no solution has been provided to the problem of providing proximity services that preserve privacy entirely within existing social networks. We develop a privacy-preserving proximity-based solution that provides both symmetric and asymmetric proximity testing entirely within social networks. We realize three different proximity-based services: KN-service (Proximity between known users), UN-service (Proximity between unknown users), TN-service (Proximity between an anonymous user and a target).

4.1 Introduction

Several social networks provide their users with proximity-based services, such as the *Nearby Friends* feature of Facebook. There are also social networks founded on geospatial features, among which, services based on the reciprocal proximity of users like Tinder, Foursquare, Jiepang, FullCircle, Gowalla, and Facebook Places. These services are exposed to serious privacy threats, because we do not have guarantees that the provider is fully trusted and enough immune to data breaches. Although proximity-based services have been deeply studied in the literature for more than a decade [111, 98, 120, 128, 85], the most severe threat model of a global passive adversary has never been analyzed. Concretely, this is the case of services entirely delivered to social-network users within the social network (due to the fact that the social network provider, playing as a global passive adversary, can monitor the flow of all the messages in the network), without assuming external communication channels.

In the paper [39], we dealt with this problem and proposed a solution that allows any pair of users to perform a *proximity test* without revealing to the adversary that the test was performed. This goal had not previously been achieved in the literature. The solution we proposed in [39] jointly solves two problems: (i) the first is to allow that proximity is detected by the social network provider without discovering the positions of the users. To achieve this, we use an approach based on *grid-tag* [181]. This measure alone is not enough to achieve our strong privacy goal. The social network provider, in fact, may observe all the messages exchanged between users and, then, discover that they are performing a proximity test. Therefore, we have to solve jointly a second problem (ii) which can be seen as an anonymous communication problem against the global passive adversary. However, we are in a specific situation in which no general-purpose connection-oriented communication is required, but only the anonymous exchange of a few short asynchronous messages. In the literature, the problem of anonymous communication in social networks has been addressed in some works [126, 88]. However, [126] only deals with anonymous group communication (therefore it cannot be used for proximity testing), and [88] does not provide sender anonymity against the global passive adversary (thus for the reciprocity of proximity testing, it would be inapplicable in our context). First of all, it is advisable to check whether the existing solutions, which have not been designed in the context of social networks, can be trivially applied to solve our problem. To do this, we need to refer to any P2P overlay anonymous routing approach resistant to the global passive adversary. Indeed, social network users can play the role of peers. Existing P2P overlay network routing techniques resisting to the global passive adversary are based either on *mixnets* [68, 93, 31], or on *buses* [78, 28, 183]. State-of-the-art mixnet-based approaches, require a high amount of cover traffic. Any (even ideal) mixnet-based approach, requires that each peer has at least three adjacent peers in the P2P overlay networks with which constant-rate cover traffic is bidirectionally exchanged [68]. Three is the minimum number because we have to mix at least two incoming sources of traffic into one outcoming flow, in order to have the exponential fan-out mechanism achieved by mixnets. As cover traffic would result in bandwidth and CPU overhead for social network users, we should limit it as much as possible. In the approach based on buses, anonymity is achieved by implementing routes (either deterministic [78, 28] or non-deterministic [183]) independent of the communication path, which senders and receivers can opportunistically exploit. With this approach, cover traffic is drastically reduced with respect to mixnets because each node has exactly one 1-hop source and, thus, only two adjacent peers, and cover traffic is one-directional. However, both deterministic (in which the fixed route is an Eulerian path passing through all the nodes) and non-deterministic approaches (in which the latency increases with hyperbolic growth on the number of nodes) are unrealistic in scenarios with a huge number of nodes like social networks.

In [38], we focused precisely on the problem of anonymous short communications over social networks and we showed that our proposal offers a good solution to the trade-off between traffic overhead and communication latency, better than the application of existing anonymous overlay routing protocols.

Next, we extended these protocols [39, 38] in paper [40], in which we developed a protocol for anonymous short communications in social networks (against the global passive adversary) and its application to proximity-based services. The paper [40] can be related to two macro-topics, traditionally not intersecting with each other: *anonymous communication networks* and *privacy-preserving proximity testing*, which is the chosen application domain. The main contribution of this extension is to jointly achieve the security requirements of anonymous communication networks (i.e., sender and recipient anonymity) and the privacy features required in the context of proximity-based services. Another meaningful innovation regards the practical application of the anonymous routing protocol to the domain of proximity-based services, which we will see in this chapter.

Since we only consider short asynchronous messages, to solve concretely the anonymity problem we could use, for example, the protocol that we have presented in [40]. In this thesis, we do not adopt the anonymity protocol presented in [40] but a simpler version published in [39]. Our approach uses the concept of fixed deterministic routes of buses to minimize cover traffic but only to hide the sender inside a small predetermined cyclic route (thus, with acceptable latency). The recipient is hidden inside a random walk outcoming the cyclic route of the sender. This allows us to achieve *communication k-anonymity* [172] for both senders and recipients.

Since we assumed that we have an anonymity protocol, we can develop a privacy-preserving proximity-based solution that provides both symmetric and asymmetric proximity testing entirely within social networks. In fact, we decided to show in this thesis the proximity protocol created in [40] in which we use a grid-based approach allowing the users to modulate the distance in which the proximity services have to be provided in order to realize three different proximity- based services: `KN-service` (Proximity between known users), `UN-service` (Proximity between unknown users), `TN-service` (Proximity between an anonymous user and a target).

4.2 Related Work

To better contextualize the problem treated in this chapter, we examine very briefly the literature, relating to the subject of our work, that has already been extensively treated in Chapter 2. The techniques proposed in the literature to implement a *privacy-preserving proximity test* can be divided into four groups[181].

Grid-based. According to this approach, a large geographical area is organized as a grid, identifying (possibly overlapping) cells of a given shape. In the most common approach, a user who wants to disclose their proximity with another user, sends the service provider the cell identifier in which they are located in encrypted form. The papers [98, 153, 154] are good examples of this category.

Tag-based. This approach leverages the spatial-temporal location tags present in a specific area to allow the service provider to detect proximity. Unpredictable and unique tags can be implemented by employing a physical infrastructure or can be obtained by capturing some environmental features, such as Bluetooth IDs, WiFi IDs, military codes in GPS, audio signals, and atmospheric gases. The papers [120, 189, 65, 125, 82, 181, 157] fall into this class of approaches. Among these, [82, 157, 181] are the most related to our proposal and reached the same goals. However, [82] requires a fully trusted authority generating users' keys. Furthermore, it is based on differential privacy, which requires much more computational effort than our approach. [157] requires an external independent secure channel to exchange some information among the users. Therefore, it does not offer a solution fully lying within the social network.

Location-anonymity-based. These techniques are based on the application of a distance - preserving transformation allowing the users to send their transformed positions to a centralized proximity detection service able to compute the reciprocal distance but not the original positions. [146] is good example of this category. However, as deeply studied in [109], the techniques based on distance-preserving transformations are prone to various vulnerabilities.

Encryption-based. The papers falling in this class of approaches face the problem mostly by using secure multi-party computation or homomorphic-based protocols [128, 85, 94, 112, 120]. In this case, two users can perform the privacy-preserving proximity test without revealing to each other and to the provider their position. At the end of the execution of the protocol, the provider does not know anything about the result of the proximity test.

To the best of our knowledge, besides those described above ([82, 157, 181]), there is no relevant work studying how to provide privacy-preserving proximity-testing in social networks. Our solution achieves this goal by applying to social networks some ideas taken from the field of anonymous routing.

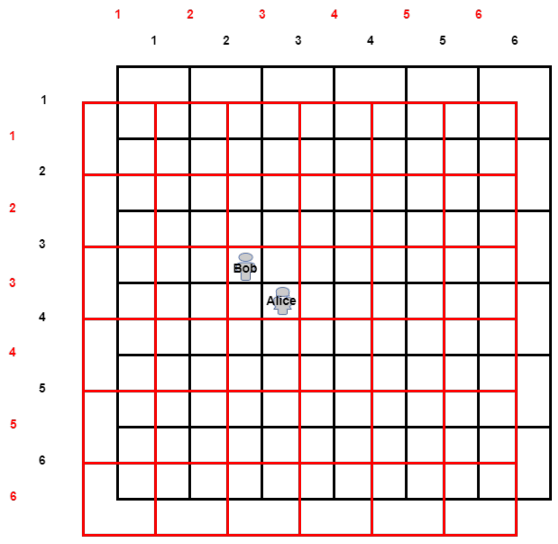


Fig. 4.1: Overlapping mechanism.

.

4.3 The Proposed Protocol

4.3.1 Grid Organization

Before explaining the protocol, we provide some preliminary notions regarding the way in which our *tag-grid-based* technique is organized and regarding the anonymity protocol we use.

Since our technique belongs to *grid-based* proximity-testing techniques [98], the entire territory is virtually divided into *cells*. We consider square cells of side a, overlapped in such a way that, at each instant, each user is simultaneously in two cells. This disposition ensures that two users at a distance less than a have at least one cell in common. We need a more sophisticated structure because we want to enable the modulation of the size of the searching area in which users want to perform their proximity test. The description of how the grid is organized has already been explained in Section 3.2. Figures 4.1 and 4.3, already explained in Chapter 3, schematize our technique.

4.3.2 Rings

In a setup phase, the users are organized into rings. A *ring* is a circular route of users of a given size k. For each ring, each belonging user is uniquely associated with a number, called *identifier*, between 1 and k. The construction of rings depends on the type of proximity service we want to provide. For example, supposing that proximity test can be performed only between users with maximum separation degree j, rings should be constructed within communities with separation not greater than j.

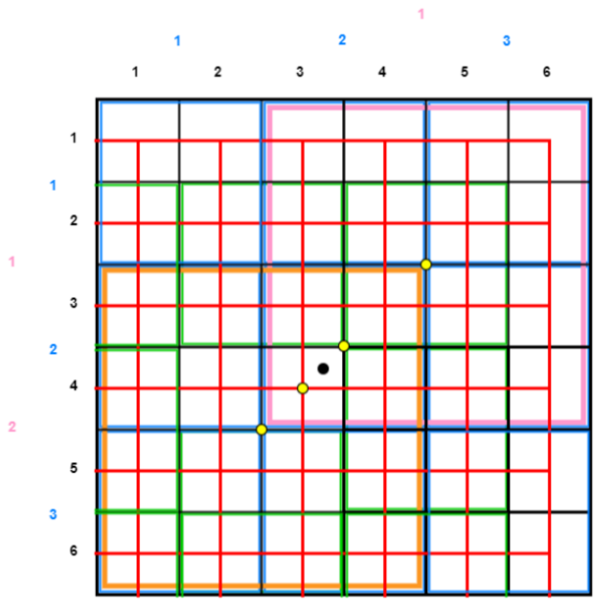


Fig. 4.2: An example of centroids.

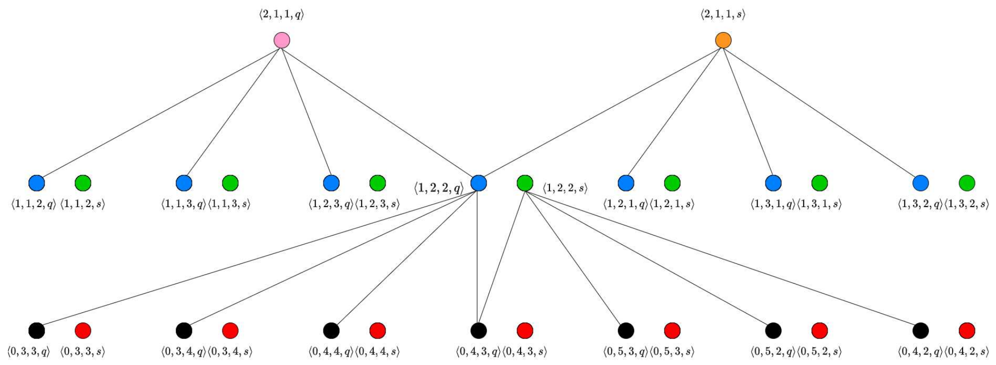

$\langle 2,1,1,q\rangle$
$\langle 2,1,1,s\rangle$
$\langle 1,1,2,q\rangle$ $\langle 1,1,2,s\rangle$
$\langle 1,1,3,q\rangle$ $\langle 1,1,3,s\rangle$
$\langle 1,2,3,q\rangle$ $\langle 1,2,3,s\rangle$
$\langle 1,2,2,q\rangle$
$\langle 1,2,2,s\rangle$
$\langle 1,2,1,q\rangle$ $\langle 1,2,1,s\rangle$
$\langle 1,3,1,q\rangle$ $\langle 1,3,1,s\rangle$
$\langle 1,3,2,q\rangle$ $\langle 1,3,2,s\rangle$
$\langle 0,3,3,q\rangle$ $\langle 0,3,3,s\rangle$
$\langle 0,3,4,q\rangle$ $\langle 0,3,4,s\rangle$
$\langle 0,4,4,q\rangle$ $\langle 0,4,4,s\rangle$
$\langle 0,4,3,q\rangle$ $\langle 0,4,3,s\rangle$
$\langle 0,5,3,q\rangle$ $\langle 0,5,3,s\rangle$
$\langle 0,5,2,q\rangle$ $\langle 0,5,2,s\rangle$
$\langle 0,4,2,q\rangle$ $\langle 0,4,2,s\rangle$

Fig. 4.3: An example of SQT.

This way, rings may represent anonymity sets. Each user belongs to just one ring. Each ring has an owner responsible for generating *tokens* (i.e., fixed-size message containers), which injects in the ring (by sending it to the next user, who, in turn, forwards the token). When a user receives a token, she/he forwards it by filling it if the user is a sender and the token is empty. Tokens are encrypted by using a probabilistic encryption scheme, so that empty and filled tokens are indistinguishable. In this way, the sender of a message is k-anonymous for a global adversary (i.e., SP).

4.3.3 The Anonymity Communication Protocol

To solve concretely the anonymity problem we could use the protocol that we presented in [40]. In this thesis, we want to focus on how to achieve proximity services guaranteeing privacy, for this reason, we assume we have an anonymity communication protocol, for example, it could be the protocol we presented in [39], which achieves the goal of anonymous communication but is a simpler and leaner version than the one presented in [40].

The approach we adopt to achieve the anonymity of communication uses the concept of fixed deterministic routes of buses to minimize cover traffic but only to hide the sender inside a small predetermined cyclic route (thus, with acceptable latency). The recipient is hidden inside a random walk outcoming the cyclic route of the sender. This allows us to achieve *communication k-anonymity* [172] for both senders and recipients.

To understand how we solved the problem of anonymous communication in social networks with protection against the social network platform (playing as a global passive adversary), suppose we have two users, Alice and Bob, as shown in Figure 4.4.

We consider a ring of size $k = 8$. Suppose Alice (associated with the red dot) wants to send a message to Bob (associated with the blue dot).

Each ring has its own *bridge user*, i.e. the user responsible for sending the messages outside the ring (playing the role of *exit user*) or for injecting into the ring the messages coming from outside (playing the role of *entry user*). The social network provider (SN) periodically sends a random number from which the users of each ring derive the value of the bridge user of the ring they belong to.

Alice waits for an empty token in her ring (recall that tokens are always turning at constant rate in the ring). Alice fills the token with the message and the value of exit node of her ring and she sends it (encrypted) to the next user in the ring. The exit node empties the token and forwards it in the ring and is responsible for routing the message outside the ring. Observe that, due to the token-based mechanism combined with probabilistic encryption, an external global passive adversary capable of analyzing all the traffic, is not able to understand who is the actual sender among the anonymity set composed of the 8 users in the ring. The exit node picks up a number q' (*hop counter*) between 1 and k. Suppose $q' = 6$. Then, it starts a random walk of length q' where the counter is decreased by one at each hop. By random walk, we mean that each node selects at random another node to forward the message. The random walk ends when the hop counter becomes 0, at the blue dot (i.e., Bob). To avoid that the global passive adversary can identify the recipient of the communication as the terminal point of the communication route, Bob starts an *inertia* random walk of $k - q'$ hops. In this example, $k - q' = 2$ (the inertia random walk involves the oranges nodes). In this portion of the route, the forwarded message is depicted with a dashed arrow to mean that it contains dummy traffic.

When the message reaches the last node, it is rebounded back (*rebound protocol*) across the route in such a way that the rebounded message reaches again Bob. This is

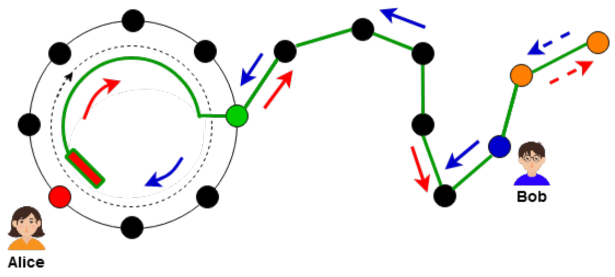


Fig. 4.4: An example of how the anonymity protocol works

.

done to avoid that a response implemented as a new communication initiated by Bob (even through a ring), if intersected with the previous route, may allow the adversary to identify Bob (i.e., intersection attack).

Now, suppose Bob wants to answer Alice. Bob includes the answer in the token and sends it across the backward path until it reaches the exit node. Finally, the latter injects the token into the ring that reaches Alice. Observe that, again, the adversary just can see a message entering in the ring, but not who is the recipient (i.e., Alice). Indeed, Alice will withdraw the response and forward the token in the ring in such a way that the reception is unobservable to the adversary.

In the figure, the red arrows define the outward path of the communication and the blue arrows define the backward path.

4.3.4 Communication Primitives

Since we have assumed that we have an anonymity protocol, we can define the three *communication primitives*: First, we informally define them, and, then, we give their definition in detail. The protocol is based on the following primitives:

- **P1: anonymous sending to explicit recipient.** This primitive is used by a sender A who wants to remain anonymous when communicating with an explicit recipient B. The explicit recipient can also coincide with the social network provider itself.
- **P2: response from an explicit recipient to an anonymous sender.** This primitive is used by the explicit recipient B of Primitive **P1** to respond to the anonymous sender A by preserving the anonymity of A.
- **P3: anonymous sending to anonymous recipient.** This primitive is used by a sender A to communicate with a recipient B in such a way that both remain anonymous. Observe that the response to this primitive can be obtained by invoking the primitive itself in the opposite direction.

These three primitives represent the building blocks of the proximity services described in Section 4.4. It is necessary to clarify that each user is associated with two *identities*. The *real identity* RI of a user is composed of three attributes: name, surname, and email address. The *SN identity* SI of a user is obtained by applying a cryptographic hash function h_R to the real identity (i.e., $SI = h_R(RI)$). SN identities are used as identifiers in the social network. Therefore, the URL of the profile of a user is derived from their SI. We assume that a user A who knows B, also knows the real identity of B and, thus, can retrieve their SN identity (needed for the communication) without leveraging SN. The autonomy of the sender in this task is necessary to maintain sender anonymity.

Now, we more formally define the three communication primitives supporting privacy-preserving services.

- **P1: anonymous sending to explicit recipient.** This primitive is invoked by a user U and receives as input a message M and a real identity RI_D. The message M (possibly encrypted) is forwarded from U to the user D with real identity RI_D by keeping U anonymous.
  U knows the SN identity SI_X and the public key PK_X of the bridge user X of the ring in which U is located. In this primitive, we say that X plays the role of *exit user*.
  First, U derives the SN identity SI_D associated with RI_D i.e., $SI_D = h_R(RI_D)$. This operation does not involve SN, so the anonymity of U is not compromised.
  Then, U waits for the earliest empty token of the ring (a user, when receives a token, has to decrypt it to decide if forwarding or processing it, because the token is encrypted with its public key) and fills its fields $\langle \bar{M}, \bar{D}, B \rangle$ as follows: $\bar{M} = M$, $\bar{D} = E(PK_X, SI_D)$ (i.e., the encryption for the bridge user X of the SN identity of the destination D), $B = 1$ (that represents the fact that the token is filled).
  We denote by T the so obtained token. Now, U encrypts the filled token T with the public key of the user Z with SN identity $SI_Z = next_A(SI_U)$. Then, the token is sent to Z. The token turns in the ring until the user X, who is the only user able to decrypt $\bar{D}$, thus obtaining SI_D.
  At this point, X forwards M to SI_D.
  Finally, X sets B to 0 and $\bar{M}, \bar{D}$ to random values and forwards the token in the ring to the user with SI equal to $next_A(SI_X)$.
- **P2: response from an explicit recipient to an anonymous sender.** This primitive is invoked by the destination D of Primitive **P1** to reply to the sender U in such a way that the latter remains anonymous. The primitive receives as input a message R (possibly encrypted). We assume as implicit input the SN identity SI_X of the bridge user X acting as exit user in Primitive **P1**.

First, D sends R to X.

X injects the response in the ring just by waiting for the earliest empty token and filling it with R. In this case, $\bar{M} = R$, $B = 1$, and $\bar{D}$ remains undefined. Then, the filled token turns in the ring until U receives R. This is the actual recipient of the response. U does not empty the token and just forwards it. This operation is done by X (for security reasons) when the token reaches them again by setting B to 0 (and the other fields to random values) and by further forwarding the token in the ring.

- **P3: anonymous sending to anonymous recipient.** This primitive is invoked by a user U and receives as input a message M and a ring identifier v_k. The message M (possibly encrypted) is forwarded from U to a user D with SN identity SI_D such that $v_k = ring(h(SI_D))$, in such a way that both U and D remain anonymous. We denote by X the bridge user, with SN identity SI_X and public key PK_X, of the ring in which U is located.
  As in Primitive **P1**, U waits for the earliest empty token of the ring and fills it by setting its fields $\langle \bar{M}, \bar{D}, B \rangle$ as follows: $\bar{M} = M$, $\bar{D} = E(PK_X, v_k)$, $B = 1$ (the token is filled).
  The token turns in the ring until the user X, who retrieves v_k.
  At this point, through the collaboration of SN, X identifies the bridge user Y of the ring v_k and forwards M to Y.
  Finally, Y injects the message in the ring as in Primitive **P2**, thus eventually reaching the actual destination D. As for Primitive **P2**, D does not empty the token, which will be emptied by Y.
  Observe that a possible reply of D to the message M sent by U can be done by using the same primitive.

Since we have explained the anonymity protocol and the communication primitives, we can present the three proximity-based services.

4.4 Proximity-Based Services

In this section we show as, through the three primitives described in Section 4.3.4, we can implement a proximity testing protocol protecting users' privacy against the global adversary. For the grid organization underlying the services, we have to consider the structure presented in Chapter 3, in Section 3.2. Recall that we use a grid-based approach allowing the users to modulate the distance in which the proximity services have to be provide.

In particular, we provide three proximity-based services. The first is a service aimed to test the proximity of users who know each other. We call this service KN-

`service` (standing for *known nearby service*). Roughly, we provide the privacy features to a service similar to *Facebook Nearby Friends*. The second service is used to test the proximity between users who do not know each other but make public some information (photos, preferences, etc.). We call this service `UN-service` (standing for *unknown nearby service*). The service allows detecting proximity of unknown users only on the basis of their agreement. This service extends the features given by services such as *Tinder*, by enabling the above privacy features. Finally, the last service regards proximity testing of a user with respect to a (static or moving) target. We call this service `TN-service` (standing for *target nearby service*). The privacy requirement is that the user remains anonymous also with respect to the target. This service extends services such as *Tripadvisor* or *BlaBlaCar*.

4.4.1 `KN-service`: Proximity between Known Users

In this section, we describe the first proximity-based service: `KN-service` We consider two users knowing each other who want to discover reciprocally if they are in proximity. The test is *symmetric*. This means that if a user X discovers the proximity of another user Y, then Y discovers the proximity of X.

The service is implemented through five phases.

The first phase is the *handshake procedure*. We consider a pair of users X and Y and we suppose the handshake is started by X. X knows the real identity RI_Y of Y. Through the Identity-based Encryption scheme (i.e., a type of public-key encryption in which the public key of a user is represented by some unique information associated with the user's identity, as explained in Section 2.3), X encrypts, under the identity RI_Y, the message $M_Q = Q||RI_X$, where Q is a random value and RI_X is the real identity of X. We denote by C_Q such an encrypted message. Then, X retrieves the SN identity of Y, denoted by SI_Y, and the ring which Y belongs to. They are obtained as follows: $SI_Y = h_R(RI_Y)$ and $v_k = ring(SI_Y)$. Given an element x, we denoted by $ring(x)$ the ring to which x belongs. Then, X invokes Primitive **P3** by passing C_Q and v_k as input. Y, who is the only user able to decrypt C_Q, retrieves $Q||RI_X$. Similarly to X, Y encrypts through IBE, under the identity RI_X, the message $\bar{M}_Q = Q||RI_Y$. We denote by $\bar{C}_Q$ such an encrypted message. Y retrieves $SI_X = h_R(RI_X)$, $v_j = ring(SI_X)$ and replies to X through Primitive **P3** by passing $\bar{C}_Q$ and v_j as input. Hence, X obtains $Q||RI_Y$ as confirmation of their request.

At this point, X (Y, resp.) generates a random value E_X (E_Y, resp.), called *Ephemeral ID*, an on-the-fly key K_X (K_Y, resp.) for the response, builds a message $M_X = E_X||Q||K_X$ ($M_Y = E_Y||Q||K_Y$, resp.), and encrypts it, through IBE, under the real identity RI_{SN} of SN. We denote by C_X (C_Y, resp.) this encrypted message. Now, X (Y, resp.) sends anonymously the message C_X (C_Y, resp.) to SN. This is done by X (Y, resp.) by invoking **P1** with input RI_{SN} and C_X (C_Y, resp.). SN, through Q, links E_X and E_Y, computes $H_{XY} = h(E_X \oplus E_Y)$, and invokes **P2** by passing H_{XY} encrypted with K_X (K_Y, resp.) as input to respond to both X and Y. This ends the handshake procedure. As a result of this procedure, the two users declare reciprocally their intention to test proximity. SN is aware of this fact but knows only the one-time pseudonymous (i.e., the ephemeral IDs) of these users and links them thanks to the value Q.

At this point, the *position notification* procedure starts. Suppose that X wants to detect the proximity of Y if they are at a distance corresponding to the level l_X and that Y wants to detect the proximity of X if they are at a distance corresponding to the level l_Y. First X (Y, resp.) retrieves the set of centroids $C_X^{l_X}$ ($C_Y^{l_Y}$, resp.) and the set of salts $R_X^{l_X}$ ($R_Y^{l_Y}$, resp.), as defined at the end of Section 3.2. For each level $0 \leq i \leq l_X$ ($0 \leq i \leq l_Y$, resp.), for each centroid $C_X \in C_X^{l_X}$ ($C_Y \in C_Y^{l_Y}$, resp.) of level i, and for each salt $R_X \in R_X^{l_X}$ ($R_Y \in R_Y^{l_Y}$, resp.) of level i, X (Y, resp.) computes the digest $h_X = h(C_X \| R_X)$ ($h_Y = h(C_Y \| R_Y)$, resp.). We denote by H_X (H_Y, resp.) the list of the obtained digests. At this point, X (Y, resp.), encrypts with IBE, under RI_{SN}, $H_X \| E_X \| \bar{K_X}$ ($H_Y \| E_Y \| \bar{K_Y}$, resp.), where $\bar{K_X}$ and $\bar{K_Y}$ are on-the-fly keys to encrypt the response. We denote by $\bar{C}_X$ ($\bar{C}_Y$, resp.) this encrypted message. Finally, X (Y, resp.) invokes **P1** by passing $\bar{C}_X$ ($\bar{C}_Y$, resp.) and RI_{SN} as input. This ends the position notification.

The next phase of the protocol, called *proximity detection*, is performed SN side. This is done by simply computing the intersection $H_X \cap H_Y$ and by checking whether this intersection is not empty. If this is the case, then SN detects that X and Y are in proximity. The intersections performed in this phase are done only between the pairs of users whose ephemeral IDs are linked through the same random Q. Furthermore, the proximity is detected only if X and Y are at a distance less than half of the length of the side of the square of the minimum level between l_X and l_Y. Suppose that $l_X < l_Y$. This means that, even though Y selects a greater distance than X, the proximity will be detected only until a distance corresponding to the level l_X. This happens because the set H_X contains only the obscured centroids detected by X until the level l_X and does not provide any information about the higher levels. Suppose now that $H_X \cap H_Y \neq \emptyset$ (i.e., proximity between X and Y has been detected). In this case, SN should report this result to X and Y.

This is done through a phase of the protocol called *proximity notification*. In this phase, SN invokes **P2** by passing the encryption with key $\bar{K_X}$ ($\bar{K_Y}$, resp.) of $H_{XY} \| H_X \cap H_Y$ as input to reply to both Primitives **P1** invoked by X and Y.

The last phase is the *ephemeral confirmation* procedure. Once $H_{XY} \| H_X \cap H_Y$ is obtained, X (Y, resp.), for each $hr \in H_X \cap H_Y$, computes $\bar{hr} = h(hr)$. We denote by H_R the list of the obtained digests. To be sure that SN does not forge a fake contact, X (Y, resp.) invokes Primitive **P3** by passing $E_X \| H_R$ ($E_Y \| H_R$, resp.), encrypted with

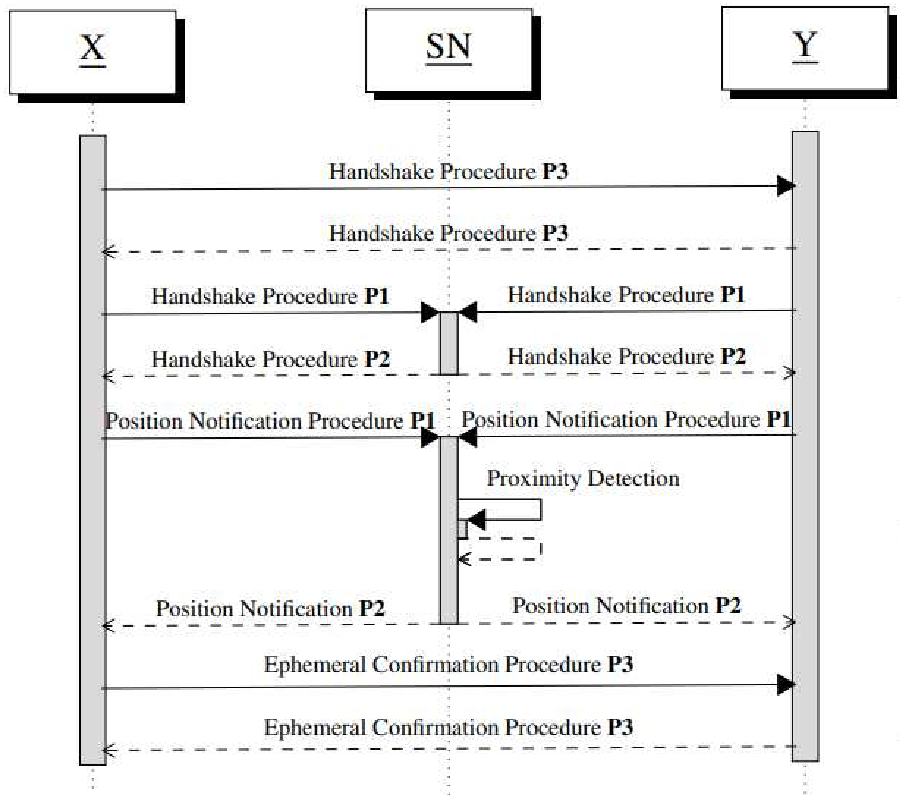

X
SN
Y
Handshake Procedure **P3**
Handshake Procedure **P3**
Handshake Procedure **P1**
Handshake Procedure **P1**
Handshake Procedure **P2**
Handshake Procedure **P2**
Position Notification Procedure **P1**
Position Notification Procedure **P1**
Proximity Detection
Position Notification **P2**
Position Notification **P2**
Ephemeral Confirmation Procedure **P3**
Ephemeral Confirmation Procedure **P3**

Fig. 4.5: Sequence diagram of the `KN-Service`

.

IBE under RI_Y (RI_X, resp.), and v_k (v_j, resp.) as input. Through E_Y and E_X, X and Y respectively, are able to compute H_{XY} and to check the correspondence with the value obtained from SN. Moreover, H_R allows them to identify the centroids of the cells they share. Specifically, X (Y, resp.) checks that the list H_R computed locally coincides with the list provided by Y (X, resp.).

The sequence diagram of Figure 4.5 shows the flow of the proximity procedure in the case of a positive result.

4.4.2 `UN-service`: Proximity between Unknown Users

In this section, we consider another symmetric case. We propose our privacy - preserving service, called `UN-service`. The users involved in the proximity procedure do not know each other, and for this reason, the preliminary handshake can not be performed. Among the current commercial systems belonging to this category, probably the best-known service is Tinder, in which unknown users can come into contact if they are in proximity and match some preferences. In particular, each user advertises a set of attributes (possibly, not identifying) and expresses a preference for the attributes of other users. To detect proximity, it is necessary that the users reciprocally show an expression of interest. Our `UN-service` consists of several phases and the whole proximity-testing activity remains unobservable by the social network provider. The first step is the *advertising procedure*. As in the position notification of the previous service, a given user X generates a list of digests H_X by using the salts $R_X^{l_X}$ and the centroids $C_X^{l_X}$ until the level l_X they want to set. At this point, X creates an *advertisement* $I_X = \langle A_X, T_X, PK_X \rangle$. A_X is the set of attributes that X wants to advertise. T_X is a pair $\langle E_X, ring(SI_X) \rangle$, where E_X is an ephemeral (random) ID generated by X and $ring(SI_X)$ is the ring which X belongs to. Finally, PK_X is a public key generated by X. For each salt $R_X \in R_X^{l_X}$, I_X is then encrypted with a symmetric key derived by R_X. We denote by C_X^R the set of all the obtained encrypted messages. Since the salts are distributed in a restricted area, the advertisement I_X can be decrypted only by the users in that area (who detect at least one common salt and, then, they can derive a valid key). At this point, X is ready to send all the needed information to SN, that is, the digests (representing the obscured position), and the encrypted messages containing the advertisement. Thus, X can send them together with an on-the-fly key K_X, which SN will use for the response. The information sent by X to SN is then: $M_X = C_X^R | H_X || K_X$, which X encrypts with IBE under the real identity of SN RI_{SN} obtaining C_X. This information is sent to SN by invoking Primitive **P1** with input C_X and RI_{SN}. This concludes the advertising procedure.

Now, suppose that another advertising procedure is performed by a user Y in proximity who sends $M_Y = C_Y^R || H_Y || K_Y$ to SN. As in the previous service, the *proximity detection* is performed SN side, and, since X and Y are in proximity, it results that $H_X \cap H_Y \neq \emptyset$. SN has to notify the proximity (and the advertisement) to X and Y. This is done through the *proximity notification* phase. In detail, SN encrypts C_Y^R (C_X^R, resp.) with K_X (K_Y,resp.) and passes the result of this encryption as input to **P2**, to reply to X (Y, resp.). At this point, X (Y, resp.) is able to decrypt at least one advertisement contained in C_Y^R (C_X^R, resp.) and retrieves A_Y, T_Y, and PK_Y (A_X, T_X, and PK_X, resp.). If both X is interested in the attributes A_Y and Y interested in the attributes A_X, then the *preference-expression* procedure starts, in which X and Y exchange with each other (through SN) their reciprocal interest. In particular, given $T_Y = \langle E_Y, ring(SI_Y) \rangle$, X computes $H_{XY} = h(E_X \oplus E_Y)$, encrypts $H_{XY} || H_X$ with IBE under RI_{SN} and invokes Primitive **P1** by passing the result of this encryption and RI_{SN} as input. A dual procedure is executed by Y.

At this point, SN performs the *preference-expression detection*. Since SN receives from Y the digest $H_{YX} = h(E_Y \oplus E_X) = h(E_X \oplus E_Y) = H_{XY}$, it detects the reciprocal expression of preference between X and Y. The result of this detection is notified through the *preference-expression notification*. Therefore, SN replies to both the users by sending $H_{XY} || H_X \cap H_Y$ through **P2** (it is encrypted, as usually, with an on-the-fly key sent by X and Y). At the end of this phase, X and Y are informed about their reciprocal interest.

At this point, a final procedure, called *preference-confirmation*, is performed. In this procedure, the users agree on a common secret, by using Diffie-Hellman with perfect forward secrecy [116], in which the public keys included into the advertisement work as authentication of the destination. Moreover, the positions are also reciprocally checked. The common secret will be used to make the successive communication confidential. Specifically, X generates a public DH parameter and encrypts it with PK_Y. We denote by CDH_X the result of this encryption. For each $hr \in H_X \cap H_Y$, X computes $\bar{hr} = h(hr)$ and obtains the list of digests H_R. Then, $CDH_X \| H_R$ and $ring(SI_Y)$ are passed as input to Primitive **P3**. Observe that both PK_Y and $ring(SI_Y)$ are contained in the advertisement of Y. On the other hand, Y performs the dual operation. At the end of this procedure, if X and Y are the real owners of PK_X and PK_Y, respectively, they are able to obtain the public DH parameter of the other user and generate a common secret that can use to communicate. Moreover, through H_R, they confirm their proximity. In detail, X (Y, resp.) checks that the list H_R computed locally coincides with the list provided by Y (X, resp.). The sequence diagram of the `UN-Service` is reported in Figure 4.6.

4.4.3 `TN-service`: Proximity between an Anonymous User and a Target

The last service regards proximity testing performed by users with respect to a given target. Unlike the previous two services, this service is *asymmetric*, in the sense that the user performing the test discovers the proximity of the target without disclosing their proximity to the target. We consider the most challenging case in which we do not allow to make public the position of all the targets on a map. This prevents from massive retrieval of target positions that could be inadequate for privacy or business reasons.

Observe that the target could be indifferently static or moving. A possible scenario that presents mobile targets is that of car sharing, in which the target are the available cars of a given company. However, to make public the positions of all the available cars on the whole map would give an improper advantage to competitors of this company by allowing easy benchmarking. Other examples of moving targets are ride-sharing, crowd-shipping, or applications supporting people to obtain aids by volunteers when they are away (consider for example the case of blind people). Instead, an example of static target is represented by points of interest (POIs). In this case, the public availability of POIs on a map may not seem like a problem. In general, this is true, but there could be some cases in which the publicity of such POIs should be limited (for example in the domain of intelligence, or sensitive tar-

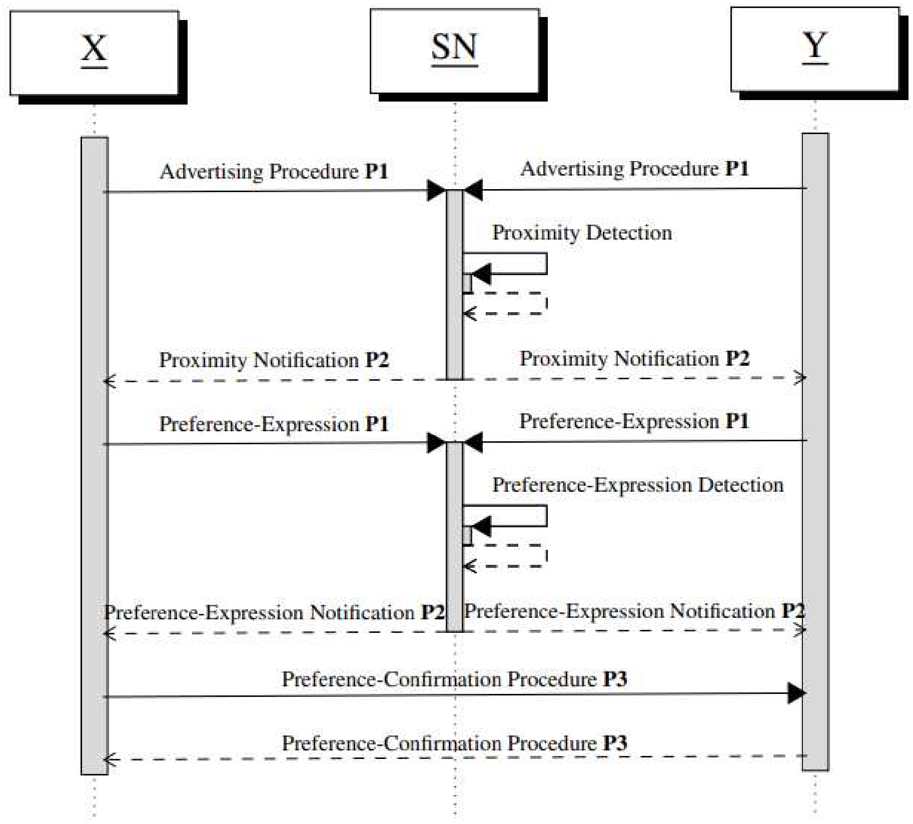

X
SN
Y
Advertising Procedure P1
Advertising Procedure P1
Proximity Detection
Proximity Notification P2
Proximity Notification P2
Preference-Expression P1
Preference-Expression P1
Preference-Expression Detection
Preference-Expression Notification P2
Preference-Expression Notification P2
Preference-Confirmation Procedure P3
Preference-Confirmation Procedure P3

Fig. 4.6: Sequence diagram of the `UN-Service`

.

gets), also for security reasons. For the above reasons, the naive solution of providing users with a map including all the targets to allow a local privacy-preserving proximity testing cannot be adopted.

Given these premises, we can describe how this service is provided. We consider a user X and a target T. T performs, periodically, an *advertising procedure* simplified with respect to that described in the previous section. In detail, the advertisement of T, say I_T, contains only an advertising message AM_T with information purpose. As no response is enabled towards T, there is no need to include in the advertisement the information used for this purpose, i.e., key, ephemeral ID, and ring identifier. To make the advertisement readable only for those users at distance compatible with their requirements, an encrypted version of I_T should be provided, for each involved level, by using the key derived by the corresponding salt, until a given maximum level l_T. As mentioned in Chapter 3, in Section 3.2, at a given level, due to tag-cell overlapping, multiple salts can be detected and, thus, multiple encrypted messages are needed. All the encrypted versions of I_T, along with the set of obscured positions H_T derived by T, are sent to SN (encrypted with IBE) through Primitive **P1**.

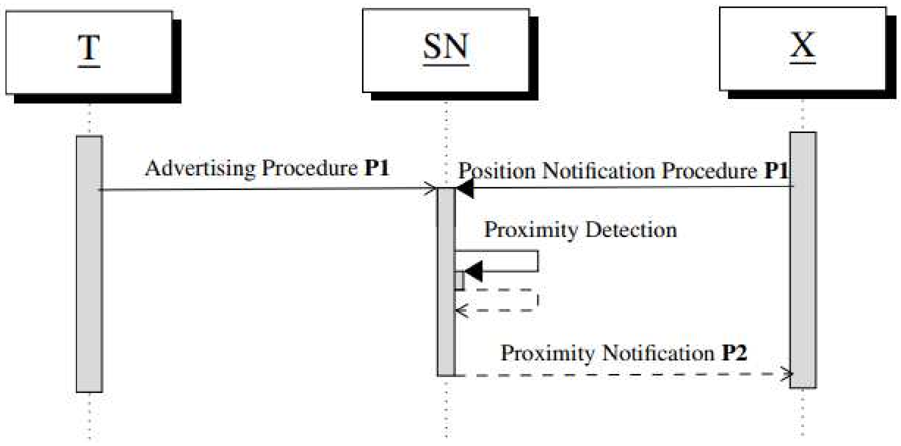

T
SN
X
Advertising Procedure **P1**
Position Notification Procedure **P1**
Proximity Detection
Proximity Notification **P2**

Fig. 4.7: Sequence diagram of the `TN-Service`

.

Due to the asymmetric nature of this service, X should not make any advertising procedure. X just generates a list of obscured positions H_X and sends them to SN through Primitive **P1** (*position notification* procedure).

As in the previous services, SN performs the *proximity detection* procedure, in which checks that $H_X \cap H_T \neq \emptyset$ and notifies the encryptions of the advertisement I_T to X through Primitive **P2** (*proximity notification* procedure).

X is able to decrypt I_T and to discover their proximity.

Observe that it is also possible to implement a client-side profile-based filtering to limit the information that the user has to process.

The flow of this proximity procedure is represented in the sequence diagram in Figure 4.7.

4.5 Security Analysis

In this section, we provide the security analysis of our solution. **Adversary Model.** In our threat model, we consider as an adversary the social network provider SN. It acts as a global passive adversary able to monitor the entire flow of messages exchanged between users within the social networks. SN is honest but curious in the sense that it legally performs the steps of the protocol, but attempts to break the anonymity of the user victim.

Our analysis could be performed at two levels: (i) the anonymity communication protocol and (ii) proximity testing. First, we could show that the anonymity communication protocol described in Section 4.3.3 allows anonymous communication between users or between a user and SN. However, in this thesis, the aspect on which we want to focus our efforts is proximity testing. Therefore, having assumed that we have an anonymity protocol, we can make the following assumption.

Assumption A1: The nodes of rings are selected in a way that the background knowledge does not allow the adversary to have more information than the uniform distribution. Therefore, rings represent *anonymity sets* of size k, in which each user is identified with probability $\frac{1}{k}$. **A1** can be satisfied if rings are built by considering suitable social network communities, depending on the kind of provided proximity service.

The only information that SN can obtain is the ring in which A belongs to and then, SN can identify A only with probability $\frac{1}{k}$, due to Assumption **A1**. Furthermore, when the message is sent from the exit node of the ring of A to B, it passes exactly through k nodes and B is one of such nodes. Therefore, the probability that SN can identify B is $\frac{1}{k}$. When B replies to A, SN does not know who is B in the path involved during the rebound protocol and knows only the exit-node of the ring, then it can identify A and B only with probability $\frac{1}{k}$, according to Assumption **A1**.

Therefore, the following theorems hold:

Theorem 4.1. *A user U invoking **P1** can be identified with probability not greater than $\frac{1}{k}$ (sender anonymity).*

Theorem 4.2. *A user U receiving a message through Primitive **P2** (as response to Primitive **P1**) can be identified with probability not greater than $\frac{1}{k}$ (recipient anonymity).*

Theorem 4.3. *Let U be a user sending a message to a user D through **P3**. It holds: (1) U can be identified as the sender with probability not greater than $\frac{1}{k}$ and (2) D can be identified as the recipient with probability not greater than $\frac{1}{k}$.*

The second level of our analysis concerns proximity testing built on top of the anonymous communication protocol. We show that the three services described in Section 4.4, leveraging the three Primitives **P1**, **P2**, and **P3**, protects the privacy of the users as specified in the next three theorems.

Theorem 4.4. *Consider two users X and Y (who know each other) leveraging the* `KN-service` *described in Section 4.4.1. It holds: (1) SN can guess the fact that X and Y are performing the test with probability not greater than $\frac{1}{k}$, (2) they detect their proximity only if they are really in proximity, (3) X discovers the proximity of Y only if Y discovers the proximity of X.*

Proof. We start with (1). Consider the handshake procedure. Due to the IBE scheme, to encrypt the messages M_Q and $\bar{M_Q}$, X and Y do not contact any PKI to obtain the public-key of the other user. These messages are exchanged through **P3**, therefore by Theorem 4.3, the anonymity of X and Y can be broken with probability not greater than $\frac{1}{k}$. Then, they invoke **P1** to send M_X and M_Y to SN that replies by sending

H_{XY} through **P2**. Since M_X and M_Y do not contain any information linkable to the identities of X and Y, due to Theorems 4.1 and 4.2, again, SN can identify them with probability not greater than $\frac{1}{k}$.

Now, consider the position notification procedure. X and Y send a list of digests obtained by applying a cryptographic hash function on a centroid and a salt that depend on the position of the user. SN cannot recover the salt that is physically distributed in the zone in which the users are located, thus the digest cannot be reversed and then the centroids cannot be identified. Therefore, the list of digests does not carry any information about the identity of the users. Since this list is sent through **P1** and the corresponding response through **P2**, again, by Theorems 4.1 and 4.2, the anonymity of X and Y can be broken only with probability not greater than $\frac{1}{k}$.

Finally, regarding the ephemeral confirmation procedure, E_X and E_Y are exchanged through **P3**. Thus, by Theorem 4.3, the anonymity of X and Y can be broken only with probability not greater than $\frac{1}{k}$.

The proof of (1) is concluded.

Regarding (2), proceed by contradiction. The statement of (2) does not hold if two cases occur: (2.1) either X or Y tries to invent a fake positive proximity result, or (2.2) SN tries to invent a fake positive proximity result between X and Y. We do not consider any other user since the other users involved in the communication see only encrypted data and cannot tamper them.

Considering (2.1), the proximity between X and Y is detected only if the lists of digests H_X and H_Y have at least one digest in common. To obtain the same digest, X and Y have to share the same centroid and the same salt. Even though the centroid can be obtained from any point of the world, the salt is physically distributed in a specific zone associated with the centroid, then the users have to be located in the same area. Regarding (2.2), the way in which SN can invent a contact is to provide X and Y with a tampered $H_{XY}||H_R$ when they are not in proximity. Anyway, the ephemeral confirmation procedure prevents such an attack since both H_{XY} and H_R can be checked by X and Y.

Finally, consider (3) and suppose that Y wants to discover the proximity of X without X can do it. The only way is to use a different ephemeral identifier, say $E_{Y'}$, during the position notification procedure such that X cannot recognize $h(E_X \oplus E_{Y'})$. Anyway, since X does not recognize $h(E_X \oplus E_{Y'})$, X does not perform the ephemeral confirmation procedure, so that Y does not know E_X, and then Y cannot detect to be in proximity of X.

Theorem 4.5. *Consider two users X and Y (who do not know each other) leveraging the* `UN-service` *described in Section 4.4.2. It holds: (1) SN can guess the fact that X and Y* *are performing the test with probability not greater than $\frac{1}{k}$, (2) they detect their proximity only if they are really in proximity, (3) X discovers that Y expresses a preference for X only if Y discovers that X expresses a preference for Y, (4) no spoofing of advertisement is allowed.*

Proof. We start by proving (1). During the advertising procedure, X and Y send M_X and M_Y to SN through Primitive **P1**. Since M_X and M_Y do not contain any information linkable to the identities of X and Y, due to Theorem 4.1, SN can identify them with probability not greater than $\frac{1}{k}$.

If X and Y are in proximity, they receive some information by SN through Primitive **P2**. By Theorem 4.2, SN can identify them with probability not greater than $\frac{1}{k}$.

Consider now the preference-expression procedure. X and Y send H_{XY} (i.e., a non-identifier), to SN through Primitive **P1**. SN replies through Primitive **P2**. By Theorems 4.1 and 4.2, SN can identify them with probability not greater than $\frac{1}{k}$.

Finally, regarding the preference-confirmation procedure, X and Y communicate between them through Primitive **P3** and then, by Theorem 4.3, SN can identify them with probability not greater than $\frac{1}{k}$. This concludes the proof of (1).

Regarding (2) (proceeding by contradiction), it does not hold if two cases occur: (2.1) either X or Y tries to invent a fake proximity between them, or (2.2) SN tries to invent a fake proximity between X and Y.

With the same reasoning as the proof of Theorem 4.4, (2.1) does not occur since X and Y have to obtain the salt. Regarding (2.2), it does not occur since the preference-confirmation procedure ensures that X and Y have at least a digest in common.

Concerning (3), during the preference-expression procedure, SN replies to X and Y only if $H_{XY} = H_{YX}$. Thus, they learn reciprocally their preference.

Finally, concerning (4), suppose Y tries to spoof an advertisement of another user Z located in the same area and uses this advertisement to enter into contact with X. Since Y does not know the public key PK_Z of Z, X and Z are unable to generate the DH secret during the preference-confirmation procedure. The proof is then concluded.

Theorem 4.6. *Consider a user X performing a proximity test with a target T leveraging the* `TN-service` *described in Section 4.4.3. It holds: (1) SN can guess the fact that X is performing the test with probability not greater than $\frac{1}{k}$, (2) X detects the proximity with T only if they are really in proximity, (3) T does not detect the proximity with X.*

Proof. Regarding (1), X sends only a list of obscured positions H_X through **P1** and receives an advertisement through **P2**. Similarly, T sends non-identifiers (through

P1) to SN. Therefore, by Theorems 4.1 and 4.2, SN can identify X with probability not greater than $\frac{1}{k}$.

Concerning (2), to decrypt the advertisement of T, X has to share the same salt. Therefore, they have to be in proximity.

Finally, regarding (3), X, during the entire proximity test, does not provide any identifying information. Therefore, there is no way for T (even collaborating with SN) to detect that X is performing the test.

This concludes the security analysis.

5

Improving Electronic Monitoring of Sex Offenders

Electronic monitoring is a valuable approach to the control of sex offenders. It also avoids prison overcrowding and protects victims. Two technologies are currently being adopted: RFID and GPS. In this chapter, we focus on this topic from a perspective that considers both the privacy of the victim and the security of the solution against the offender's misbehavior. The theoretical analysis shows that GPS is the best choice when high-security requirements are desired. In fact, radio frequency attacks are possible for RFID, endangering the victim. However, when GPS is adopted, privacy issues become critical. In particular, when considering a victim moving around the territory, it is unacceptable to track them even with the goal of offering protection. To overcome this drawback, we propose a GPS-based solution that does not allow the victim's location to be revealed unless the offender is nearby, thus finding a solution that advances the state of the art.

5.1 Introduction

In recent years, due to the increasing number of crimes committed and the overcrowding of prisons, many efforts have been devoted to identifying the appropriate legal measures to address the aforementioned problems. Also in this scenario, the advancement of technology has made it possible to solve various problems and to elaborate new types of solutions. In the case of crimes in which the safety of a victim is threatened by a sex offender, if the offender's imprisonment is not applied, alternative measures are applied, such as restraining orders. According to a restraining order, the sex offender cannot get closer to the victim (or areas of potential victims) more than a certain distance. These alternative measures can profitably exploit technology [19, 175]. In particular, electronic monitoring (EM) is used to monitor the movements of the offender according to the order issued by the court, thanks to the use of a device worn by the offender (and the victim) [124]. The reasons for the use of electronic monitoring remain diverse, such as providing humane and affordable sanctions, reducing prison crowding, or avoiding the construction of new prisons.

Through the use of electronic monitoring, it is also hoped to suppress the criminal behavior of offenders monitored.

Obviously, one of the biggest and most immediate advantages is that the police can constantly check whether the offender is complying with the restraining order imposed by the law [178, 66, 29]. Two different types of restrictions can be imposed, *inclusion* or *exclusion* zones. An inclusion-zone order defines the area in which the offender must be located. For example, this order may be applied in combination with the house arrest measure. Exclusion zones define the zones in which the offender should not enter. A typical exclusion zone is the victim's home. We refer, in cases like this, to *static* exclusion zones. There can also be a *dynamic* exclusion zone when the restraining order prohibits the offender from approaching the victim beyond a certain distance.

EM technology has advanced over time. The initial systems were only able to determine if a tagged offender had strayed beyond a certain distance from their home. Subsequently, Radio Frequency (RF) technology was joined by Global Positioning Systems (GPS) that allow more sophisticated monitoring, thus allowing offenders to be monitored at much greater distances and at any time of the day [29]. Therefore, in this context of localization tracking technologies, Radio Frequency (RF) and Global Positioning System (GPS) are the reference technologies [130]. They can also be used in combination. In both cases, monitoring is under observation by the law enforcement agency (LEA). This means that the LEA can verify that the offender complies with the area indicated by the restraining order.

In this chapter, we analyze the problem of EM in the general case, by identifying a gap in existing techniques in the case of dynamic exclusion zones, which is the most critical case from the victim's privacy perspective. The problem we deal with falls within the field of *proximity-based services*, in which the proximity of two users or the proximity of a user to a given target should be detected in order to provide a given service. The general analysis is done by defining a theoretical framework (see Section 5.3) in which functional, security, and privacy requirements are jointly considered. From the analysis, it emerges that for all types of EM, except in the case of dynamic exclusion zones, current approaches are appropriate. In contrast, for dynamic exclusion zones, functional requirements would be securely achieved only with GPS-based approaches. However, unlike the case of static exclusion zones, serious privacy problems arise regarding the victim. The fact that GPS should be used for dynamic exclusion zones but cannot be used for privacy reasons is a shortcoming of current systems that we want to overcome in this chapter. Indeed, we propose a privacy-preserving GPS-based solution that does not allow the victim's location to be revealed unless the offender is nearby, thus finding a solution that advances the state of the art.

5.2 Related Work

In recent years, the increase in cases of domestic violence or stalking has led to finding new strategies to protect victims. Among the various adopted measures, many states have approved the use of electronic monitoring (EM) technologies to supervise sex offenders. Electronic monitoring is a term that refers to various location control technologies that allow the management of offenders or prisoners awaiting trial [74]. Indeed, EM has always been used primarily to remove criminals from detention, either as an alternative to incarceration or a means of post-release supervision. However, EM is adopted to monitor people also in other contexts [61, 117, 131].

The use of technology to supervise is not new. It is in the US that this practice has been initiated [143]. In [161], the authors show the evolution of electronic offender-tracking systems, whose origin dates back to the 1960s, thanks to the study conducted by Ralph Kirkland Gable and William S. Hurd at the University of Harvard [150]. In the mid-1960s, early technologies were tested on groups of parolees, released mentally ill patients, and research volunteers. However, these devices were large, difficult to hide, and impractical to wear on a daily basis [127]. In 1987, about 900 people participated in nationwide electronic monitoring programs in more than US 21 states [149]. In 1998, that number increased to over 95,000 [92]. By the end of the 2000s, EM was being used for violent offenders [58]. In 2006, 22 states passed legislation requiring or authorizing the use of Global Positioning Satellite (GPS) technology to track sex offenders. In 2009, there were more than 200,000 GPS and radio frequency (RF) monitoring devices in use across the United States and the State court system [46].

Europe has also accumulated a body of experience in the field of EM of sex offenders [123, 121, 122, 148]. For an overview of EM use around the world, see [124].

Wearable devices are typically used to track people, especially in the context of e-health [131, 84]. In the context of Electronic Monitoring of sex offenders, two technologies are currently being used, namely *Radio Frequency* (RF) systems and *Global Positioning Systems* (GPS) systems. The latter, from a qualitative analysis conducted by the authors of [45], has shown to be more effective. In this chapter, we provide a formal validation of the above claim.

RF-based technology allows checking if an offender is located in a specific place (typically the offender's home) through signals at a reasonable frequency. Differently, GPS-based systems allow the collection of the actual location data of offenders thanks to a device worn by the offender. This way, the monitoring center can in real-time track the movements of the offender.

In the literature, GPS technology is used in three different ways: *active GPS tracking*, *passive GPS tracking* and *hybrid systems*. Active systems are those systems that transmit real-time information about the offender's location to an almost real-time monitoring center. Differently, passive systems collect location information at a specific time interval and are sent in an aggregated way to the monitoring center. Finally, hybrid systems combine both passive and active monitoring capabilities, where data is sent after a longer interval of time with respect to the active system.

The current technologies aim to allow an alternative to imprisonment, reduce non-compliance and violations of supervisory conditions, assist offenders in reintegration into society, prevent future and repeated sexual offenses by convicted offenders, and increase public safety. The EM technology used today has greatly improved compared to the past and allows for an increasingly sophisticated use as a tool for the management, punishment, and public protection of the offender. Furthermore, current devices are generally smaller and can be hidden. However, there are still limitations [105]. A first limitation is represented by the battery life that some providers have tried to solve by developing a portable charging pack that can be clipped on to the electronic anklet. A problem that has not yet found a solution is that of jammers that can be used to block or interfere with both the RF and the GPS signals. Although it could be possible to identify when a jammer was used, there is a risk that a crime will be committed before the jammer is identified, or that the offender has escaped at that time. In addition, even the hardware can be damaged or removed, although there are mechanisms that allow notifying the competent authority in the event of tampering. This does not prevent offenders who have successfully removed their label from offending or fleeing.

In the context of Electronic Monitoring of sex offenders, a new problem has emerged. The offender is strictly prohibited from approaching the places usually frequented by the victim. If the offender approaches the victim or the pawn in places where there is no prohibition, the electronic bracelet would be of no use. The device is used exclusively to monitor the movements of an individual within one or more predefined places, generating an alarm only when the offender accesses the areas that are excluded. Therefore, some countries, such as Spain and Italy [108, 17], are adopting a new approach based on *proximity tracking*, and it is the scenario in which the victim is also equipped with an electronic bracelet to detect the presence of the offender in the immediate proximity. As a matter of fact, the above problem can be related to the issue of *proximity-based services*, in which the proximity of two users or the proximity of a user to a given target should be detected in order to provide a given service [141, 111, 128, 39, 75, 34].

Our chapter focuses on this kind of monitoring, which we call *dynamic exclusion zones*, as the exclusion zone is virtually defined as an interdiction zone around the victim, thus moving with them. We improve the state of the art by enabling GPS in this kind of monitoring yet preserving victims' privacy. This way, we obtain the increased robustness of the GPS solution with respect to the RFID, without paying the intolerable price of continuous victim tracking.

5.3 Background, Threat Model, and Problem Formulation

Electronic Monitoring is a tool widely used for more than a decade to enforce restrictions on the movements of offenders, according to the order of a court. The order may regard *inclusion* or *exclusion* zones. When an inclusion zone is imposed on the offender, they are enforced not to leave this zone. In the case of exclusion zones, the offender is not allowed to enter these zones. Examples of restriction of the first type are the *house arrest measure*, or the restriction limiting the movements of the offender within the area of a given city. Examples of the second type are the denial for the offender to approach certain places, such as schools or sport-clubs for pedophiles, or to approach the house of the victim (and other places attended by them too) for general sex offenders. In the latter case, also the denial to approach the victim, independently of the place in which they are placed, is also possible.

Therefore, we can identify three different *security requirements*, which we denote as:

1. **SIZ**: inclusion zones, which are necessarily static;
2. **SEZ**: static exclusion zones;
3. **DEZ**: dynamic exclusion zones.

The technologies used to implement EM are basically two (possibly used in combination):

- **Radio Frequency (RF)**: The offender is equipped with a tamper-proof tag worn on an arm or on an ankle, which plays as radio frequency transmitter. The tag is active and the signal has an action range of the magnitude of 100 meters. Sometimes tags offender-size are two-pieces.
- **Global Positioning System (GPS)**: GPS is the geostationary satellite system allowing GPS devices to detect continuously its position coordinates and, thanks to an on-board SIM card, to communicate them to a server. In this case, the GPS device is a tamper-proof tag worn on an arm or on an ankle of the offender. The

coordinates are sent to the provider of the service, which we assume to be a telephone service provider (TSP) – as it happens in Italy, and/or the law enforcement agency (LEA).

Let us see how the two technologies can be used to achieve the above security requirements. For each security requirement, we consider the two technologies.

- **SIZ** (static inclusion zones).
  - **SIZ with adoption of RF**. In the case of RF, the inclusion zones should be tagged with non-tamperable and not removable RF receivers equipped with a SIM card. The restriction imposed on the offender is verified if the signal is received. In absence of a signal, the restriction is considered violated. RF is suitable for SIZ only for limited areas. In practice, this system is adopted for house arrests.
  - **SIZ with adoption of GPS**. In the case of GPS, the inclusion zones are trivially identified by TSP/LEA and, then, the respect of the restriction can be checked by computing the distance between the tracked coordinates of the offender and the borders of the zones.
- **SEZ** (static exclusion zones).
  - **SEZ with the adoption of RF**. In the case of RF, the exclusion zones should be tagged with non-tamperable and not removable RF receivers equipped with a SIM card too. The restriction imposed on the offender is verified if the signal is not received. If the signal is received, then, the restriction is considered violated. RF is suitable for SEZ only for limited areas. In practice, this system is adopted to guarantee that the offender keeps far from the house of the victim or their office.
  - **SEZ with adoption of GPS**. In the case of GPS, the exclusion zones are trivially identified by TSP/LEA. Similarly to the case of SIZ, the respect for the restriction can be checked by computing the distance between the tracked coordinates of the offender and the borders of the zones.
- **DEZ** (dynamic exclusion zones).
  - **DEZ with adoption of RF**. In the case of RF, the only difference with respect to SEZ, is that the receiver is a portable removable device kept by the victim.
  - **DEZ with the adoption of GPS**. In the case of GPS, the system could be implemented by equipping also the victim with a GPS tracker, and, then, by continuously computing the distance between the coordinates of the victim and the coordinates of the offender. However, in most countries, a similar measure would be unacceptable from a privacy point of view [176, 177].

We formalize in more detail the above security requirements in terms of security policies and, then, we identify the realistic threat model under which the above policies should be able to guarantee the safety of the victim.

We schematize the policies by defining the following predicates.

- SI is the predicate stating that an RF signal is received by the receiver.
- GP is the predicate representing the reception of the GPS coordinates TSP/LEA-side.
- DI is the predicate stating that the tracked position of the offender is inside the inclusion zones in the case of GPS.
- DO is the predicate stating that the tracked position of the offender is outside the exclusion zones in the case of GPS.
- AL is the predicate representing the alarm state, which results in some intervention from the side of the LEA (i.e., call to the offender, call to the victim, arrival of the police at the victim and at the offender).
- OK means that no violation is detected so that the state of the victim is assumed to be safe.

From a logical point of view (as it will be clear later), it can be realized that the closed world assumption can be adopted, so that $\neg AL$ cannot be considered in general equivalent to OK, and vice versa.

The security policies are the following. We distinguish the security policies per adopted technology (RF and GPS).

- **SIZ with RF technology.**
$$\begin{aligned} \neg SI &\rightarrow AL \\ SI &\rightarrow OK \end{aligned} \tag{5.1}$$
Indeed, if the RF signal is not received by the receiver, then the alarm is generated, because the offender is assumed to be outside the inclusion zone. Conversely, no anomaly is detected if the signal is received.
- **SEZ and DEZ with RF technology.**
$$\begin{aligned} SI &\rightarrow AL \\ \neg SI &\rightarrow OK \end{aligned} \tag{5.2}$$
Indeed, if the RF signal is received by the receiver, then the transmitter is close to it (close to the victim, in the case of **DEZ**), thus within the exclusion zone. Therefore, the alarm is generated. Conversely, if no signal is received, the offender is assumed to be outside the exclusion zone. Therefore, no anomaly is detected.
- **SIZ with GPS technology.**

$$\begin{aligned} \neg DI &\rightarrow AL \\ DI &\rightarrow OK \\ \neg GP &\rightarrow AL \end{aligned} \tag{5.3}$$

The first two implications just state that when the GPS coordinates of the offender are received, the alarm is generated if those coordinates are not within the inclusion zone. Conversely, no alarm is generated if the detected location of the offender is within the inclusion zone. The last implication refers to the case in which the offender's device does not transmit any coordinate. In this case, the alarm is raised.

- **SEZ and DEZ with GPS technology.**

$$\begin{aligned} \neg DO &\rightarrow AL \\ DO &\rightarrow OK \\ \neg GP &\rightarrow AL \end{aligned} \tag{5.4}$$

The first two implications just state that when the GPS coordinates of the offender are received, the alarm is generated if those coordinates are within the exclusion zone. Conversely, no alarm is generated if the detected location of the offender is outside the inclusion zone. The last implication refers to the case in which the offender's device does not transmit any coordinate. In this case, the alarm is raised.

Let us consider now our **threat model**. We can assume that *spoofing* and *impersonation* are not possible. Therefore, the messages sent by the offender can always be assumed to come from a legal device, and they cannot be tampered. Moreover, we can assume that the offender-side device and the RF receivers (except from DEZ) are not removable. However, we cannot assume that the offender (playing as an attacker) is not able to inhibit the transmission of the RF/GPS-coordinates messages. Indeed, a *jamming attack* [100, 118, 155] is always possible. Consider that, the absence of a GPS signal could also derive from physical obstacles (thus not from malicious behavior of the offender).

Under the above threat model, these are the possible attacks/anomalies to consider:

- RAT: an attack performed by the offender on the RF transmission;
- GAT: an attack performed by the offender on the GPS transmission;
- GFA: the case in which the GPS signal is obscured by an accidental physical obstacle.

We can write the following implications to formalize the above attacks/anomalies.

$$RAT \to \neg SI$$
$$GAT \to \neg GP$$
$$GFA \to \neg GP$$

First, consider the case of RF, and, then, the RAT attack. According to 5.1, for **SIZ**, $\neg SI \to AL$. Then $RAT \to AL$. This is a good behavior of the model.

Instead, for **SEZ, DEZ**, again from 5.1, we have that $\neg SI \to OK$. Then, $RAT \to OK$. This represents a serious vulnerability of the security policy in the considered threat model because the general objective of the protocol (the safety of the victim) is not reached. Indeed, the case of RAT can reasonably coincide with a physical attack of the offender on the victim.

Now, we test whether the security policies stated below in the considered threat model guarantee the safety requirement in the case of GPS.

Here, the attacker can perform a GAT attack. In this case, $GAT \to \neg GP$, but, according to 5.3 and 5.4, for all the requirements (i.e., **SIZ**, **SEZ** and **DEZ**), it holds that $\neg GP \to AL$. Therefore, $GAT \to AL$. Similar reasoning can be done in the case of GFA, and for GFA we have the same implication as GAT (i.e., $GFA \to AL$).

Therefore, the security policy is then safe, because no safety failure occurs, even though some false alarms are possible (i.e., in the case of GFA).

From the analysis above, we should conclude that, in the case of sex offenders, in which the security requirements are **SEZ** and **DEZ**, RF is not adequate and we should adopt GPS.

This is certainly possible for **SEZ** because the GPS coordinates of the exclusion zones are static, but for **DEZ**, it would require victim tracking, which is not acceptable from the privacy point of view and not compliant with the most legal systems [176, 83, 13].

On the other hand, **DEZ** is much more effective than **SEZ**, since it allows permanent protection of the victim, also when they move from predetermined locations (such as home and office).

The aim of this chapter is to understand how we can adopt a GPS-based solution without facing the above privacy and legal issues. Obviously, any privacy-preserving GPS-based solution can be combined with an RF solution to have a backup system, but, from a logical point of view, it does not increase the security of the electronic monitoring solution. Indeed, if the attacker is performing simultaneously RAT and GAT, RF cannot help us to distinguish the case of GAT with exclusion violation on the victim from the case of GAT without exclusion violation (that could drastically change the urgency of the police intervention). Similarly, RF cannot help us to distinguish the case of *GFA* with exclusion violation from the case of *GFA* without exclusion violation for the same reason.

5.3.1 Grid-based Approach

The solution proposed in this chapter is grid-based [98, 153, 154]. Also in this scenario, we take advantage of the *tag-grid-based* approach presented in Section 3.2. Now, we resume the proposed structure adapting it to the context dealt with. Therefore, the territory is organized as a grid composed of *cells* of a certain shape (squares, hexagons, circles, etc.). We consider square cells overlapping each other covering the entire territory in which the victim and offender move. This structure is at the basis of the mechanism allowing the detection of the proximity between offender and victim at a distance less than that allowed by the restraining order.

Furthermore, since different restraining orders may require different distances, we implement a hierarchical grid organization including different levels, to allow distance modulation. Higher levels correspond to higher distances. Through this organization, the law enforcement agency may set the distance in the electronic monitoring system to be compliant with the restraining order. This is done by selecting the proper level in the hierarchy.

More formally, we design a hierarchical spatial index based on the concept of *quad tree* [40], in which also overlapping is enabled at each level. We call this structure *shifted quad tree* (SQT). A quad tree is a tree in which each internal node has exactly four children. It can be used to partition a 2-dimensional area into regions of different sizes. Specifically, the entire area is associated with the root of the tree and it is partitioned into four regions, each associated with a child of the root. Recursively, each region is partitioned into four regions, and so on. The smallest indexed regions are associated with the leaves of the tree.

We denote by 0 the level corresponding to the leaves of the SQT and by k the level corresponding to the root (i.e., k is the maximum level).

We start by describing the level 0, by referring to Figure 5.1.

Consider two grids (one black and the other red) composed of square cells, initially coinciding and shifting the red one across the left-bottom diagonal by the half diagonal of the square. The result is depicted in Figure 5.1. Each cell (black or red) corresponds to a leaf of the SQT and it is identified by the coordinates of its center. We denote such a pair of coordinates as *centroid*.

It is easy to see that this structure satisfies two properties: (1) the victim/offender is always simultaneously within two cells and (2) if the distance between the victim and the offender is less than half of the length of the side of the cell, then they have at

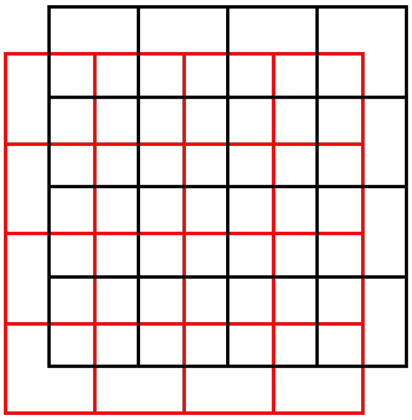


Fig. 5.1: Level 0 of the SQT.

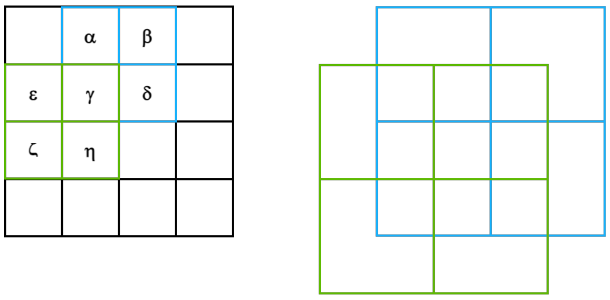


Fig. 5.2: Level 1 construction of the SQT.

least one cell in common. Therefore, the level 0 corresponds to the minimum safety distance equal to half of the side of a cell.

To enable greater distances, we leverage the same shifted structure at higher levels. To understand the mechanism, we show as the level 1 of the SQT is implemented.

Consider just the black grid of level 0, reported for clarity in Figure 5.2. The four adjacent black cells $\alpha, \beta, \gamma, \delta$ are aggregated into a cell blue of level 1, say A. At this point, we take the four black cells $\epsilon, \gamma, \zeta, \eta$ of level 0 and aggregate them into a green cell of level 1, say B. Observe that B can be viewed as the cell A shifted across the left-bottom diagonal by the half diagonal of the square (similarly to level 0). The above procedure is applied to the entire level 0, by taking (four by four) the other adjacent cells, thus completing the level 1.

In the end, we obtain two grids, one blue, and the other green shifted in the same way as the black and red grids, but with greater size of the cells (twice the size of the cells of level 0).

At this point, the same procedure can be iterated to build the level 2 starting from the blue grid, and so on, until the desired roots (to have a forest of SQTs).

The SQT-based grid organization of the territory will be exploited in our protocol to identify the proximity of the victim and offender. This is done by enabling the periodical sending from them of the centroids of the cells they belong to (once the level of the SQT is set). The server will detect the proximity in the case of common centroids.

However, allowing the sending of centroids as plaintext would disclose the position of the victim. This would be intolerable from a privacy point of view. Even sending a cryptographic hash of the centroid is not resolutive, because the size of the domain of centroids is not enough large to prevent the reverse of the digests. To avoid this, we combine our grid-based approach with a tag-based mechanism, to make available an unpredictable value, associated with a point in space and over time used as a *salt* when digests are generated. This way, reversing is prevented.

In the literature [120], by tag we can refer to Bluetooth IDs, Wifi IDs, military codes in GPS, audio signals, LTE, and atmospheric gases.

In our protocol, we choose to implement the salt mechanism by relying on the collaboration of a telephone service provider (TSP), which transmits the salts through the cellular network. In this approach, we exploit cellular cells to identify a region of the space in which, for a given time interval, a random salt, with a suitable rounding protocol, is periodically sent in broadcast to all the devices belonging to this cell. We organize the tag-mechanism in a hierarchical way, where level 0 is represented by the physical TSP cells and, at higher levels, we set suitable cells (by aggregating underlying cells) that we call *virtual* TSP cells. With the term *tag-cell*, we generically denote either physical or virtual TSP cells. Specifically, tag-cells of level k are obtained by aggregating tag-cells of level $k-1$, as described in the following.

Before discussing it, we deal with the problem of possible misalignment, at each level of the hierarchy, between the grid and the tag-cell structure.

Indeed, it may happen that a grid cell is cut from a tag-cell. In this case, two users belonging to the same grid cell could receive two different salts, thus compromising proximity detection.

To avoid this, we implement a mechanism to obtain tag-cells that include entirely grid cells of the same level, thus preventing the above drawback. This is obtained with levels greater than 0 by construction, just by aggregating tag-cells of a lower level to obtain tag-cells of the successive level that include entirely grid cells of the same level.

At level 0, we have to face the problem that physical TSP cells have a fixed size that cannot modulate to adapt to our solution. To solve this problem, at level 0, we introduce a tailored mechanism.

We assume that the grid cells of level 0 have a size of the same order of magnitude as the physical TSP cells or they are smaller. This is not an abstraction since TSP cells currently deployed on the territory have at least a coverage area of at least 200 meters (picocells).

At level 0, our mechanism works as follows. Each TSP antenna broadcasts:

1. A *primary salt* in its coverage area (physical TSP cell), and
2. the primary salts of the adjacent TSP antennas. These salts are referred as *secondary salts*.

Then, each user receives a set of salts of level 0. It is easy to realize that, two users in the same grid cell of level 0 receive at least one salt of level 0 in common.

For a generic level l, we need that the same salt is provided in the entire tag cell of level l. To do this, the TSP broadcasts the same salt of level l in all the physical TSP cells forming the tag cell of level l.

To conclude this section, we summarize the information captured by the users and exploited in the protocol presented in Section 5.4.

Consider the offender S and the victim V at a distance less than half of the length of the side of the cells of level l.

1. S detects two centroids C_1^S and C_2^S and V detects two centroids C_1^V and C_2^V such that at least one between C_1^S and C_2^S coincides to one between C_1^V and C_2^V.
2. S detects a set of salts of level l R_S and V detects a set of salts of level l R_V such that $R_S \cap R_V \neq \emptyset$.

The importance of the above claim will be clear in Section 5.4.

5.4 The Proposed Protocol

In this section, we propose our solution to the problem formulated in Section 5.3. First, we define all the involved actors, and then we describe the protocol on which the solution is based.

The actors are:

- *Victim V*: the person who is under the threat of a sex offender.
- *Sex Offender S*: the person who threatens the safety of the victim and who is, therefore, under police surveillance.
- *Law Enforcement Agency L*: the central entity that monitors S and is authorized to collect and handle data and information about them. No monitoring threatening privacy is allowed to L regarding the victim, except in case of emergency.

As discussed in Section 5.3.1, in addition to the above actors, also the *Telephone Service Provider* (TSP, for short) has a role in the solution, consisting of the periodical sending of salts in the territory.

To implement our solution, we leverage the SQT structure introduced in Section 5.3.1. Suppose L received a court restraining order specifying d as the minimum distance that the S must maintain with respect to V. According to this distance, L sets the proper level l of the SQT. Specifically, l is selected as the minimum level such that, denoting by x the length of the side of its cells, it holds that $x \geq 2d$.

Now, we describe how the solution is implemented by considering separately the offender-side equipment and actions and the victim-side equipment and actions.

Offender. The sex offender will be equipped with a special portable GPS tracking device running our application and embedding also a SIM card. The device is tamper-evident and includes non-accessible memory areas. As already happens in adopted electronic monitoring systems, the device is worn on the ankle (or on the arm) of the offender and is equipped with a battery unit.

The following information is inserted by L at the set-up phase into the non-accessible memory area:

- ID_S: An ephemeral identifier of the sex offender.
- ID_V: An ephemeral identifier of the victim.
- l: The level of the SQT selected by L.

Furthermore, some information is periodically received by the SIM card and the GPS receiver and allows the application to compute further information to send to L via cellular communication, leveraging a cryptographic hash function h.

Specifically:

- The SIM card receives all the salts transmitted by TSP in that location.
- The application selects the salts associated with level l. Denote by R_S the set of these salts.
- The GPS receiver obtains the coordinates G_S identifying the position of S.
- The application computes, starting from G_S, the centroids C_1^S and C_2^S of the cells of level l which S belongs to.
- For each salt R and for each centroid C, the application computes a digest $H = h(R||C||ID_V||ID_S)$. We denote by H_S the set of the so computed digests.
- Finally, S sends $\langle H_S, G_S \rangle$ to L.

The sequence of the actions performed offender-side is summarized in the sequence diagram of Figure 5.3.

Victim. The victim should not be provided with dedicated devices. They can use their personal smartphone equipped with our specific application.

The same information stored in the offender's device is inserted by L at the set-up phase into the smartphone of the victim:

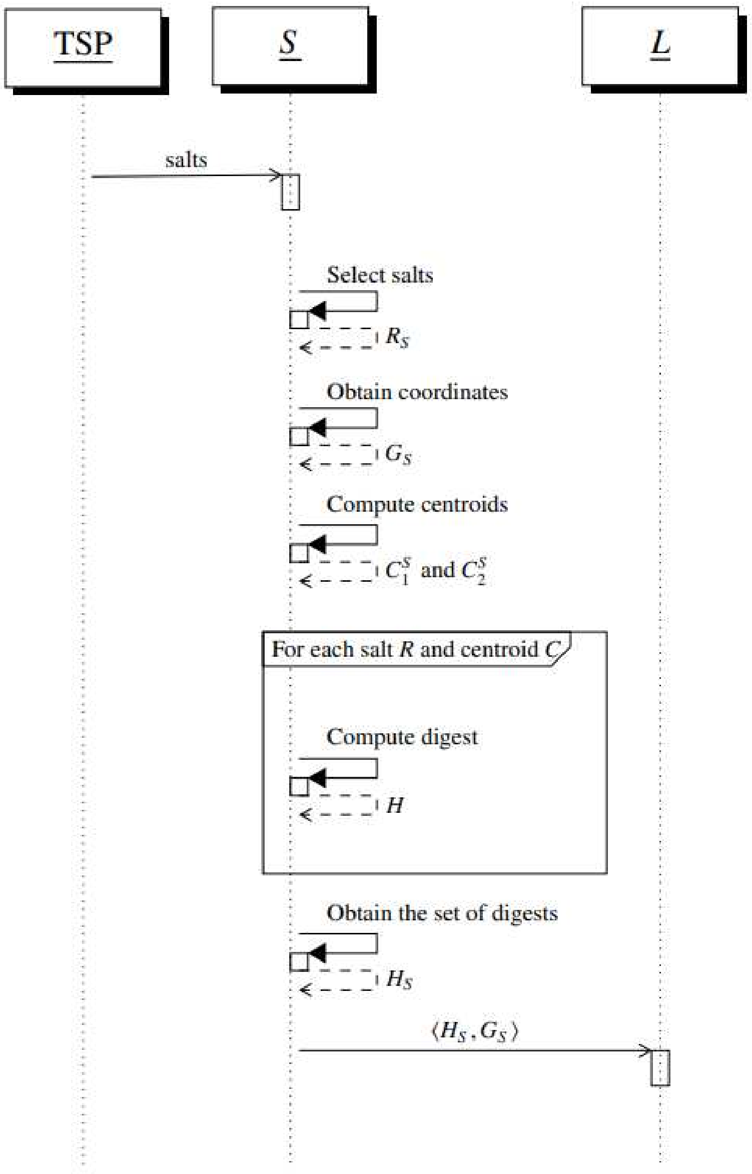


Fig. 5.3: Sequence diagram of the offender-side actions.

- ID_S: An ephemeral identifier of the sex offender.
- ID_V: An ephemeral identifier of the victim.
- l: The level of the SQT selected by L.

The application runs a sequence of actions very similar to those executed offender-side:

- The SIM card receives all the salts transmitted by TSP in that location.
- The application selects the salts associated with level l. Denote by R_V the set of these salts.
- The GPS receiver obtains the coordinates G_V identifying the position of V.

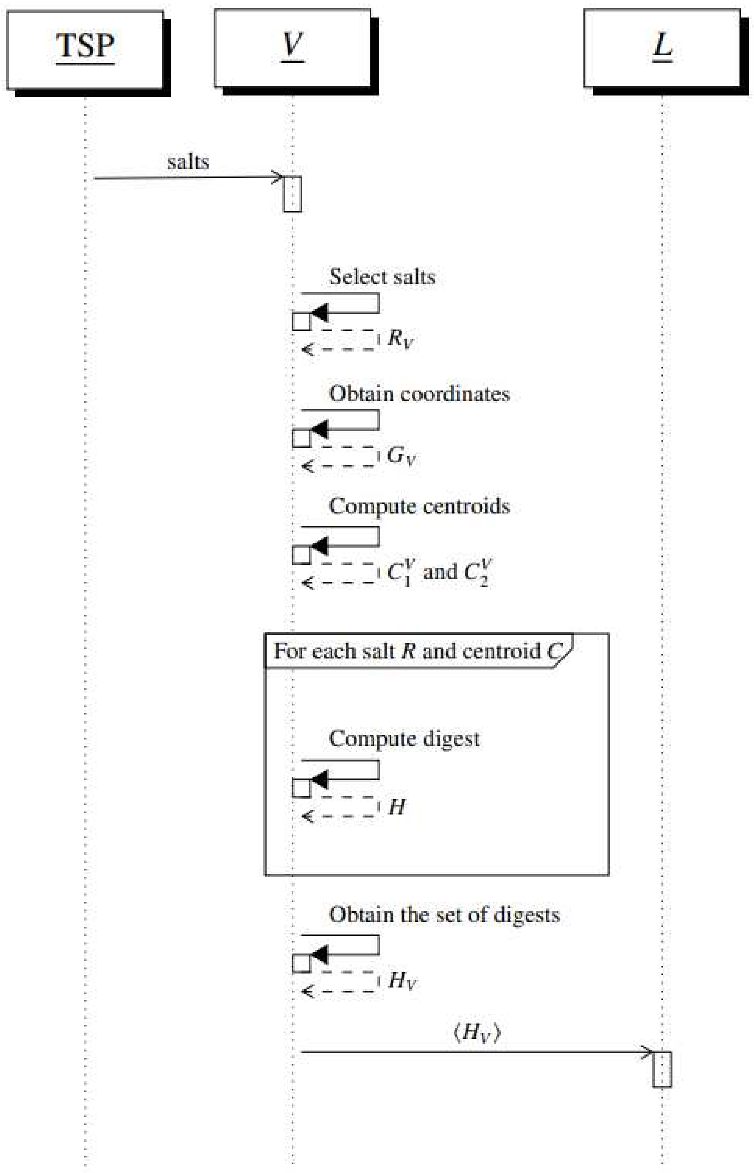


Fig. 5.4: Sequence diagram of the victim-side actions.

- The application computes, starting from G_V, the centroids C_1^V and C_2^V of the cells of level l to which V belongs.
- For each salt R and for each centroid C, the application computes a digest $H = h(R||C||ID_V||ID_S)$. We denote by H_V the set of the so computed digests.
- Finally, V sends $\langle H_V \rangle$ to L.

The sequence of the actions performed victim-side is summarized in the sequence diagram of Figure 5.4.

Observe that V does not send the GPS coordinates G_V to L, but only the non-reversible digests. This preserves their privacy.

At this point, some actions are performed server-side by L.

Law Enforcement Agency. L receives $\langle H_S, G_S \rangle$ from S and $\langle H_V \rangle$ from V.

Then, it performs as follows:

- L computes $H_{SV} = H_S \cap H_V$.
- If $H_{SV} = \emptyset$, then no action is needed and L waits for the next tuples from S and V.
- Otherwise (i.e., $H_{SV} \neq \emptyset$), the proximity between V and S is detected. Therefore, the following actions are performed:
  - L, who knows V's mobile number, sends the alarm to V and also communicates the exact location of S so that V can move away in the opposite direction.
  - Once the alarm is received, the app of V will respond with an acknowledgment and provide its exact G_V location to facilitate the possible intervention of the police that can reach both V and S.

The sequence of the actions performed Law Enforcement Agency-side is summarized in the sequence diagram of Figure 5.5.

5.5 Security Analysis

Through this section, we discuss the security guarantees offered by our proposal.

We refer to the notation introduced in Section 5.4.

Our analysis is performed in terms of *Security Properties*.

SP1: If V and S are at distance less than d, an alarm is triggered by L.

This property refers to the *correctness* of the protocol.

We show that this property is guaranteed by our solution.

Indeed, regardless of the distance between V and S, V sends L the tuple $\langle H_S, G_S \rangle$. Since S is equipped with a non-tamperable device, such a tuple may not be correctly provided only if the GPS receiver is not able to detect the GPS signal. We observe that, as discussed in Section 5.3, the absence of a GPS signal can occur in two cases. Either the offender performs a jamming attack on the GPS signal or the GPS signal is obscured accidentally by obstacles. However, in both cases, if L does not have the information from S, the alarm is triggered.

Now, consider the case in which the tuple $\langle H_S, G_S \rangle$ is correctly provided by S to L. We recall that d is the distance selected by the court order. Moreover, L selects a level l of the SQT corresponding to cells of size x, such that $x > 2d$. As illustrated in Section 5.3.1, this implies that two users at a distance less than $\frac{x}{2}$ share at least a cell. Therefore, if V and S are at a distance less than d, then they share at least a centroid, say C.

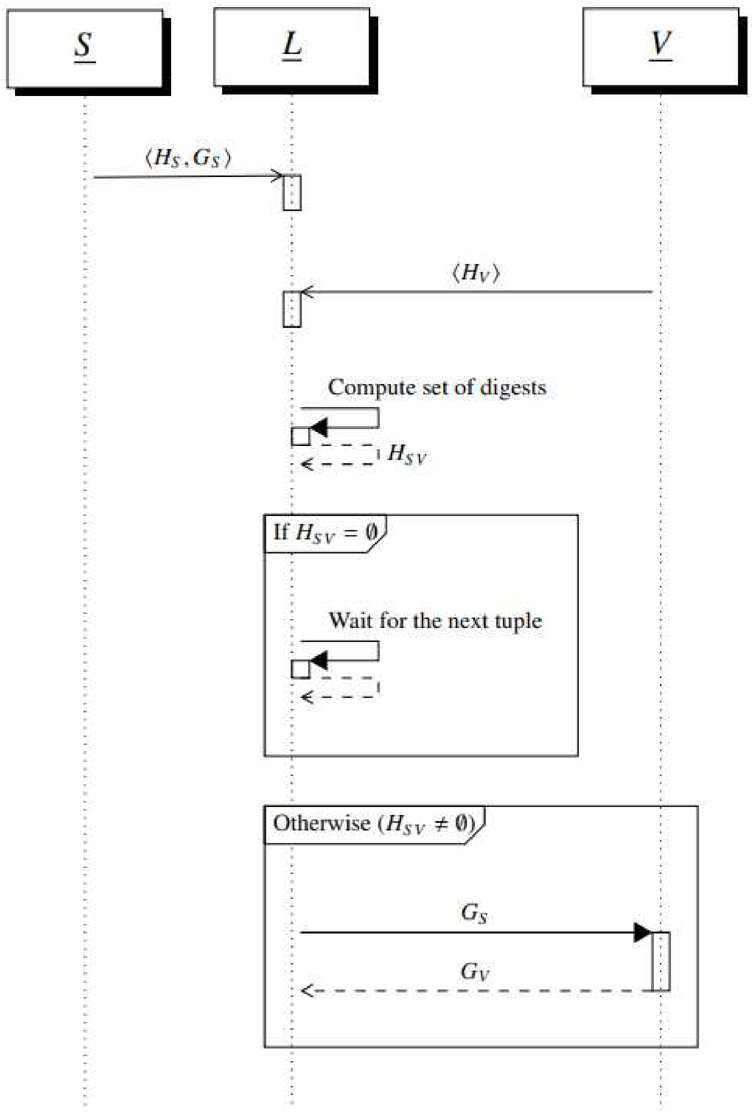


Fig. 5.5: Sequence diagram of the Law Enforcement Agency-side actions.

Furthermore, the introduction of the tag-cells ensures that V and S receive at least one common salt, say R, since they share a cell. Then, they compute the same digest $H = h(R\|C\|ID_V\|ID_S)$. Therefore, the set $H_{SV} = H_S \cap H_V$ includes at least H. This condition triggers the alarm.

SP2: L knows the position G_V of V if and only if their proximity with S is detected.

This property refers to the *privacy* feature offered to V. In our threat model, we assume L is honest-but-curious, in the sense that it performs legally the step of the protocol, but attempts to leak the privacy of V.

We show that this property is guaranteed by our solution.

Concerning the **if-part**, by definition of the protocol, when the proximity between V and S is detected, V voluntarily provides their coordinates G_V to L.

Consider now the **only if-part**. According to the assumption, L is honest-but-curious. Therefore, we assume L attempts to discover the position of V from the information provided by them. The only information V sends L is the set of digest H_S. Each element $H \in H_S$ is in the form $H = h(R\|C\|ID_V\|ID_S)$. From H, L may attempt to discover C, as it represents approximate information about the location of V.

Even though L knows ID_V and ID_S, the presence of the salt R prevents dictionary attacks performed on the domain of the centroids. Therefore, L is not able to reverse the hash function and detect C.

6 A Centralized Contact-Tracing Protocol for the COVID-19 Pandemic

The pandemic emergency brought out the importance of the concept of proximity between people and many strategies have been used to counter and restrain the spread of COVID-19. This Chapter focuses on the topic of digital contact tracing (DCT), in particular proximity tracing, as an important, effective and privacy-preserving measure for curbing the spread of pandemic disease outbreaks. As DCT systems can handle highly private information about people, great care must be taken to prevent misuse of the system and actions detrimental to people's privacy, up to and including mass surveillance. We propose a new centralized DCT protocol, called ZE2-P3T (Zero Ephemeral Exchanging Privacy-Preserving Proximity Protocol), which relies on smartphone localization but does not give any information about the user's location and identity to the server. Importantly, the fact that no exchange of ephemeral identities among users is required is the basis of the strong security of the protocol, which is proven to be more secure than the state-of-the-art approach, i.e., DP-3T/GAEN.

6.1 Introduction

Pandemics such as COVID-19 present large-scale crises that require collective action at the societal level to be contained, including the use of digital technologies. Digital contact tracing (DCT), in particular proximity tracing, is an important, effective, and privacy-protecting measure to stem the spread of pandemic disease outbreaks [144]. Contact tracing is a public health practice to identify and notify those people who had contact with an infected person during their contagious period. Conventional and manual contact tracing relies on people's recollection and does not scale well in episodes of rapid viral spread [64]. DCT should be considered as a complementary task with respect to traditional contact tracing because it is able to identify contacts that escape the investigation activities carried out by contact tracers (for example, whether they regard contacts with people unknown to the index case). In addition, the specific characteristics of COVID-19 infection (variable symptoms, frequent asymptomatic carriers, and incubation times relatively short) require faster detection of at-risk contacts than traditional contact tracing. DCT represents one of the weapons that information technology can provide to fight the pandemic, along with other digital weapons pursuing different goals, such as early diagnosis [167, 25] or early warning [129]. The role of digital contact tracing to control the spread of the COVID-19 pandemic has recently been studied [87, 171]. Many countries adopt DCT systems that use Bluetooth Low Energy (BLE) proximity measurements (e.g., [10, 12, 11]) and rely on the exchange of ephemeral identifiers, which are pseudo-random self-generated numbers designed to be unlinkable to each other and with the real identity of the user. In the European Union, the prevailing protocol is DP-3T (Decentralized Privacy-Preserving Proximity Tracing) [166], typically implemented via GAEN (Google-Apple Exposure Notification) [20]. DP-3T/GAEN is decentralized (i.e., does not delegate contact detection to a server) and does not utilize localization systems (such as GPS) to detect proximity, but only BLE. On the other hand, localization-based approaches lead to concerns about privacy, as witnessed by the recent act of the European Union that disapproves the use of GPS [60].

Concerns about privacy still remain an open issue, due to the fact that DCT handles sensitive information about people, and any system misuse or action harmful to 's people's privacy (up to mass surveillance) should be prevented [30]. Privacy concerns are the main reason people are hesitant to use DCT systems. Beyond the risk-based perspective, according to which the most current solutions could be considered secure enough (and thus promoted among citizens to contrast the pandemic), any step ahead in terms of security and privacy in the context of DCT is desired. A number of vulnerabilities have been reported about DP-3T/GAEN [23, 170, 26], which can lead to break protocol integrity and users' privacy. A number of specific concerns about privacy regard GAEN [97], about data that are transmitted to the back-end servers of the Google ecosystem. Privacy and security problems can be serious obstacles to the massive adoption of DCT. On the other hand, the effectiveness of DCT depends strictly on this factor. If the system is not adopted by the largest number of people, then it may give very limited benefits to the fight against the pandemic. This chapter arises precisely from these motivations. We address this research question with the aim of defining an alternative approach that overcomes the most drawbacks of DP-3T/GAEN. To understand this, we start with the observation that the basis of most of the attacks reported in the literature on DP-3T/GAEN is that, through BLE, smartphones exchange ephemeral identifiers. They are pseudonyms, but uniquely associated with individuals, and not under the exclusive control of the individuals with whom they are associated. It is rather intuitive that, to avoid the exchange of ephemeral identifier, a centralized approach could be adopted in which a server collects the positions of smartphones and, thus, detect the proximity of individuals. However, this solution would be much worse from the privacy perspective, because the server would continuously track people.

In this chapter, we face the challenge of centrally detecting the proximity of individuals, thus avoiding the exchange of ephemeral identifiers, without providing the server with any information that enables it to track people. The goal we pursue is to design a centralized approach that, unlike those existing in the literature, does not provide the server with tracking capabilities. Consequently, since the exchange of ephemeral identifiers is no longer necessary, the drawbacks of DP-3T/GAEN are thus solved and no new security and privacy issues are introduced, resulting in a solution definitely better than DP-3T/GAEN.

To achieve the above goal, we rely on the availability of an effective localization system (the technologies for this are already mature and very near advances are expected with 5G and 6G). The proposed centralized protocol is called ZE2-P3T (*Zero Ephemeral Exchanging Privacy-Preserving Proximity Protocol*). Unlike DP-3T/GAEN, the proposed protocol does not rely on the exchange of (even pseudonym) identities. Despite the use of localization, by using some cryptographic obfuscation mechanisms, our protocol does not allow the server to track people. Moreover, since ZE2-P3T does not use Bluetooth, users are not exposed to existing Bluetooth vulnerabilities [8, 63]. We prove that ZE2-P3T is more secure than DP-3T/GAEN.

The original idea underlying this chapter has been presented in [41] in a much simpler way. In this chapter, we propose a solution that strongly extends [41], and which we presented in [43]. We reduced the workload server-side by moving the whole risk computation from the server to the client and we extensively deal with the security analysis of the proposal. Specifically, we provide a theoretical framework that allows us to *measure* the security improvements over DP-3T/GAEN in a quantitative way and across a multidimensional domain, composed of three dimensions: (1) attacker's type, (2) target's type, and (3) range of the attack. Therefore, unlike the empirical and brief security analysis provided in [41], we prove formally the security of our approach by comparing it with the two designs of DP-3T, and by also giving a measure of the security gap between the protocols. Moreover, the empirical framework usually adopted in the context of DCT, based on the enumeration of existing attacks, is also preserved and contextualized within the above theoretical framework. Differently from [41], we provide the implementation of the main modules of the solution as research demonstrators (`https://github.com/vincenzodeangelisrc/ZE2-P3T`) that can be used as building blocks for a software system that fully implements our solution. Moreover, we exploited the implemented prototype to test the performance of our solution, by verifying the feasibility of the server-side computation.

6.2 Related Work

One of the most difficult challenges that modern society has ever faced is the COVID-19 pandemic. To counter and slow down the spread of the virus, new ways, new strategies, and solutions are being sought every day, in every sector, from the economic to the medical, from the political to the technological. Precisely, in the latter field, researchers are investing their effort to propose, digital solutions for tracing contacts that preserve privacy and that comply with current regulations. Existing protocols and applications can be classified in many different ways, for example, on the basis of the model on which they rely on. The model defines how the server is used and which data are required (or stored) by it. There are three possible models (1) decentralized, (2) centralized, and (3) hybrid [14, 158].

Decentralized Approaches. Several solutions decide upon a decentralized approach (such as DP-3T [166] and the Google and Apple Exposure Notification System [20]) to guarantee high privacy properties. The protocols that adopt the decentralized architecture move the main features to the users' devices. Obviously, in this case, the server is minimally involved in the contact tracing process. The most relevant protocol, among the decentralized models, is certainly the Decentralized Privacy-Preserving Proximity Tracing (DP-3T) [166]. For this protocol, two designs are defined (i.e., *Low-Cost* and *Uninkable*). In both designs, the protocol is based on ephemeral pseudonyms (called *EphIDs*) sent via BLE which are registered by nearby users. In Section 6.3, we will see in detail the operating principle of this approach. Apple and Google [20] have teamed up to realize, on the respective Operating Systems (iOS and Android), an implementation of Low-Cost DP-3T, called *Google Apple Exposure Notification* (GAEN). Several apps leverage the GAEN APIs, such as the SwissCoviD [11] app released for pilot tests by the DP-3T team, the Corona-Warn-App [67] (released in Germany), the Italian Immuni app [7], and the Aurora [51] project.

Bluetooth-based decentralized systems, such as systems that rely on DP-3T/GAEN, are vulnerable to the Paparazzi attack (which we will explain in Section 6.7) and therefore can be exploited for mass surveillance [152]. In [23], the authors unveiled two decentralized systems, based on BLE, named Pronto-B2 and Pronto-C2. These systems appear to be more resistant than DP-3T against mass surveillance attacks. Both systems can optionally be implemented using blockchain technology, but Pronto-B2 is designed to be more efficient and practical. A research collaboration led by MIT has developed the Private Automated Contact Tracing protocol, named PACT (East-coast) [145], which allows the user to also store extra metadata, such as location information, to increase the accuracy of the system. Researchers from the University of Washington proposed PACT (West-coast) protocol (Privacy-sensitive protocols And mechanisms for mobile Contact Tracing) [50]. Compared with PACT (East-coast), PACT (West-coast) saves storage space by storing fewer seeds than the PACT (East-coast) app. PACT (West-coast), like all decentralized protocols, is also susceptible to linkage and enumeration attacks [14]. Another decentralized system is CAUDHT [36], which is based on distributed hash tables and blind signatures. The entities involved are users, a distributed hash table (DHT), and the Health Authority (HA), which has the role of confirmation of positive cases. TCN (Temporary Contact Numbers) [55] is a decentralized protocol based on Bluetooth, which, to solve the scalability problem, switches from purely random TCNs to TCNs generated deterministically from some seed data. The price it pays for greater scalability is a reduction in privacy because the TCNs derived from the same report can be linked together. The Covid-Watch app [4] follows the TCN protocol.

Furthermore, there are several decentralized DCT proposals based on IoT [164, 81], blockchain [134, 24, 16], or both [69]. The system proposed in [164] can be configured to support different models, ranging from the fully decentralized to the fully centralized ones. Note that most of decentralized approaches use Bluetooth. It is important to highlight that switching on the Bluetooth interface of smartphones, can make the devices vulnerable to a number of attacks, also tailored to GAEN-based digital contact tracing, as shown in [63].

Centralized Approaches. Several solutions choose a centralized approach, such as NTK [163] and ROBERT [47] which have been developed inside Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT) [162]. In general, the advantage of centralized approaches is to provide epidemiologists with more useful data, thus allowing more effective actions to be taken to defeat the virus, but some scholars fear that these systems could become a more intrusive massive surveillance tool for governments [169]. PEPP-PT NTK is a proximity tracing system, based on BLE [163]. ROBust and privacy-presERving proximity Tracing protocol (ROBERT) is jointly developed by researchers at INRIA (France) and Fraunhofer (Germany) and has been implemented in the French Stop-Covid app [9]. Just like DP-3T, NTK and ROBERT are based on ephemeral pseudonyms sent via BLE that are registered by nearby users, with the difference that the secret keys for calculating EphIDs are created and handled by a back-end server and not from the user's smartphone [15]. The Altuwaiyan et al. solution, called EPIC, [18] is always based on Bluetooth technology, and offers a fine-grained human-to-human contact tracing scheme with hybrid wireless and localization technology. EPIC introduces a matching method that uses homomorphic encryption to match common wireless devices between the infected and the other users. However, the system can suffer from serious privacy losses and above all, it has scalability problems [56]. TraceTogether (Singapore) [12] and CovidSafe (Australia) [5] are based on the Bluetrace protocol [27]. The two apps have many similar features but differ mainly in the lifetime of the EphIDs, as Trace-Together uses a value of 15 minutes, while CovidSafe uses a value of 2 hours and, for this reason, CovidSafe is more vulnerable to replay attacks [168]. In [135], the authors propose a solution that resists replay attacks and, if location data are present, to relay attacks (see Section 6.7). Some vulnerabilities of TraceTogheter can be found in [54].

Hybrid Approaches.

Some approaches combine the characteristics of centralized and decentralized architectures, such as DESIRE [48], ConTra Corona [33], and EpiOne [165]. However, solutions based on Bluetooth technology present several vulnerabilities [8], also exposing the contact-tracing apps to DoS attacks, as recently shown in [63]. A solution to mitigate DoS attacks is presented in [52].

Another possible classification can be done on the basis of the capability of the system to trace the user's position. Indeed, location data can be useful for epidemiologic analysis and also to capture the fact that COVID-19 can also be transmitted through common environments or commonly touched surfaces [90] (*indirect* contacts). Obviously, knowing a user's location could come at a very high price in terms of privacy. All the systems considered so far do not collect exploitable users' location information. However, despite privacy issues, this is a class of techniques well represented in the literature.

User-location-aware approaches. In this class of techniques, some approaches combine Bluetooth technology with GPS, and others rely only on GPS. Hamagen [6] is based on decentralized architecture and it does not rely on logging encounters with other users in proximity via Bluetooth. Instead, it cross-checks (locally on the user's smartphone) the GPS history of the smartphone with the historical-geographical data of the cases identified by the Ministry of Health. If the application discovers that a user has been in the same place and at the same time as a diagnosed case, a notification is displayed on the user's phone. However, Hamagen is vulnerable to flaws in centralized data pool protection [99]. A similar principle is followed by the COVID Safe Paths app [3]. The Aarogya Setu [1] is based on centralized architecture and uses both Bluetooth and GPS. This app also collects location data (GPS coordinates) and self-assessment data. [142] offers a solution on how to make contact tracing centralized based on GPS data to preserve user privacy. The system uses a central party (HA) and applies multi-party computation (MPC) to achieve privacy. However, these solutions are not scalable [56], due to the computational overhead required by the adopted cryptographic protocols (i.e., MPC).

Our solution starts from the above reference framework, with the aim to overcome the privacy and security issues of current decentralized solutions. Our approach is centralized and is based on privacy-preserving absolute position detection. Concerning the location-aware class of methods, although our protocol requires that users send their localization information to the server, this information cannot be exploited by the server for anything but contact detection. This happens because the information is sent in an obscured form. Note that, our solution only relies on localization technologies and does not require the use of Bluetooth. Therefore, it does not suffer from the threats described in [63]. Concerning localization technologies, we do not make a specific choice, by assuming that a precise indoor-outdoor technology is available. This assumption is well-founded. Since smartphones are equipped with GPS, WLAN, gyroscope, accelerometer, magnetometer, and other sensors, it is possible to achieve high-precision localization. As we will see in more detail in Section 6.4, the scientific literature clearly shows that these technologies are already sufficiently mature to offer a precision (also seamless indoor-outdoor [107]) which is much higher than that required for DCT, and much higher than that provided by BLE proximity measurement [96]. Moreover, the immediate next future with 5G and, then 6G, will allow even more precision in smartphone localization techniques [35].

The novelty of our approach with respect to the existing literature is that our protocol, differently from the other proposals based on the centralized model and in favor of privacy, does not give to the server any information besides the fact that some pairs of random numbers (associated with humans) are close to each other somewhere in the territory. The specific novelty is that this is done without using complex cryptographic protocols and, thus, with no server-side computational overhead. Another important difference of our solution with respect to the relevant state of the art is that Bluetooth is not required. This implies that, as observed earlier, users' smartphones are not exposed to the threats related to the usage of this interface. But the real significance of the proposal strictly concerns the security aspects of the protocol (apart from technological aspects), as it is clearly shown in Section 6.7. Indeed, we show that, in terms of security and privacy goals, our solution outperforms the state-of-the-art protocol based on the decentralized model, which is DP-3T/GAEN.

6.3 Background and Motivations

As mentioned in Section 6.2, the DP-3T protocol [166] represents currently the prevailing approach, especially in the European Union. Although DP-3T, likewise to TCN [55], suffers from some severe disadvantages concerning users' privacy, it is the main reference approach because it is the state-of-the-art protocol adhering to the decentralized model, which is preferred to the centralized model. Apple and Google, in April 2020, released the Google Apple Exposure Notification (GAEN) framework to facilitate the implementation of BLE-based contact tracing applications. GAEN is basically an implementation of the *Low-cost* version of DP-3T. Several countries (such as Italy, Switzerland, Germany, and Latvia) have chosen applications based on GAEN (Immuni app [7], SwissCovid [11], CoronaWarnApp [67], Apturi Covid [2], respectively) as the official contact tracing system. Therefore, it is certainly salient to describe in detail how DP-3T-based solutions work. The basic idea is to install an app on each smartphone and to use Bluetooth to interact with other nearby devices to register the contacts. Hence, the actors involved in the DP-3T system are:

- The *users* in possession of a communication device (a smartphone equipped with Bluetooth running the DP-3T app).
- The *back-end server*, which acts exclusively as a communication platform and does not perform any processing aimed to identify contacts. Moreover, the privacy of users in the system does not depend on the actions of this server.
- The *health facility*, which is responsible for informing patients of the positive test results, allows uploads from phones to the back-end and determines the contagious window.

  Both *back-end server* and *health facility* are assumed to be TTPs (Trusted Third Parties), but they cannot be considered independent because they could be part of the same National Health System.

The app broadcasts an ephemeral pseudo-random ID that represents the user and also records pseudo-random IDs observed by devices in the immediate proximity. After obtaining the approval of the health facility, if a user is tested positive for COVID-19, they may upload some anonymous data from their smartphone to a central server. Before uploading, all data remains exclusively on the user's smartphone.

The DP-3T model provides two decentralized proximity tracing designs: the first, denoted as *Low-cost*, is a lightweight system at the cost of limited tracing of infected patients, the second, denoted as *Unlinkable*, offers extra privacy properties with a small increase in bandwidth. The first solution reveals minimal information to the back-end server. Each smartphone generates an initial random daily key SK_t for the current day t and, every day rotates the secret day key SK_t by calculating

$SK_t = H(SK_{t-1})$, where H is a cryptographic hash function. Each smartphone uses the secret key SK_t during the day t to locally generate a list of ephemeral identifiers ($EphID$)s that change frequently (every epoch with length l minutes). Therefore, at the beginning of each day t, each smartphone generates locally a list of $n = (24 \cdot 60)/l$ new $EphID_i$s to be transmitted during the day t. Given the secret day key SK_t, each device calculates $EphID_1 \| ... \| EphID_n = PRG(PRF(SK_t, broadcastkey))$, where PRG is a stream cipher, PRF is a pseudo-random function, and *broadcast key* is a fixed and public string. The $EphID_i$s are transmitted in random order and each $EphID$ is transmitted for l minutes. The $EphIDs$ are broadcasted via BLE announcements (the system relies on BLE beacons, whose payload is 16 bytes, which technically limits the size of the $EphID$s). These $EphID$s are then locally stored (jointly with the corresponding proximity, the duration, and an approximate indication of the time) by the other nearby smartphones. Each smartphone stores the SK keys it has generated in the last 14 days and the same happens for all the data and the $EphID$s observed and generated. A user who tested positive, only after obtaining authorization from the health authority, may send to the back-end the key SK_t and the day t corresponding to the first day on which it was considered contagious. The back-end collects the pairs (SK_t, t) of the infected patients and periodically sends them to all the other smartphones in the system. Given the key SK_t, everyone can calculate all the ephemeral identifiers $EphID$s used by the infected patient starting from the corresponding day t. The device determines the owner's risk score using the risk-scoring algorithm with its local records corresponding to the infectious $EphID$.

The second design (i.e., Unlinkable) offers better privacy properties at the cost of a greater volume of downloads and storage space required by the smartphone. The solution utilizes a Cuckoo filter [62]. The ephemeral identifiers of positive individuals are hashed and stored in a Cuckoo filter [62], which is distributed to the users of the system. The smartphone draws a random 32-byte per-epoch seed ($seed_i$) and generates the ephemeral identifier $EphID_i$ for each epoch i: $EphID_i = TRUNCATE128(H(seed_i))$, where H is a cryptographic hash function, and $TRUNC$ $ATE128$ truncates the output to 128 bits. Smartphones store the seeds corresponding to all past epochs in the last 14 days. For each observed $EphID$, the smartphone stores the hashed string $H(EphID\|i)$, the proximity, the duration, and an approximate indication of the time. The relevant aspect is that, unlike the previous solution, when a user is tested positive, they can choose the set of epochs I for which they want to reveal their identifiers. In other words, they can selectively decide which identifiers they want to communicate to the server. After the user's decision, the smartphone loads the set of pairs $(i, seed_i)$. Periodically, the back-end creates a new Cuckoo filter F and, for each pair $(i, seed_i)$ loaded by an infected user, inserts

$H(TRUNCATE128(H(seed_i))||i)$ into the Cuckoo filter F and sends this filter to all the smartphones in the system. The filter allows each smartphone to determine if the user has been in contact with an infected person.

DP-3T suffers from several attacks (which will be described in detail in Section 6.7) that can compromise user privacy and potentially lead to undetectable mass surveillance attacks [23]. This problem is the consideration from which we start as the motivation of our paper. In fact, our paper tries to offer a new declination of the centralized model overcoming the security and privacy drawbacks of DP-3T, without introducing risks usually associated with centralized digital contact tracing at feasible computational cost for the server.

To give the flavor of our approach, we give an overview of how our solution works. Also in this scenario, we adopt the tag-grid-based approach presented in the previous chapters. Therefore, the territory is virtually divided into overlapping square microcells, whose size is of the magnitude of the safety distance. Thanks to microcell overlapping, we guarantee that if two users, say Alice and Bob, are at distance less than the safety distance, they occupy at least one shared microcell. Each microcell is represented by its centroid, so that Alice and Bob, autonomously (i.e., with no reciprocal interaction), identify at least one common centroid. Each smartphone sends the server, periodically, an ephemeral identifier along with the coordinates of the centroids of all the microcells which it belongs to. Ephemeral identifiers, as in DP-3T/GAEN, are changed according to a rounding protocol. However, whilst in DP-3T/GAEN the smartphones store the ephemeral identifiers of the encountered contacts (in such a way that anybody can be informed about past at-risk contacts), in our protocol, smartphones do not exchange the ephemerals with other users. Therefore, Bluetooth is not exploited by our solution (with positive impacts on security). Observe that the fact that the smartphones send the coordinates to the server is not an issue from the privacy point of view, because these coordinates are sent in a *salted hashed* form, by using a salt broadcast by the telephone service provider and rounded periodically. This way, the server cannot know where the centroid is located, because the salted hashed coordinates are not reversible and are always different. Once an infection is reported, the server is able to detect all the contact at risk, by finding groups of ephemeral identifiers associated with the same salted hashed centroid within the same time slot. Indeed, this means that the smartphones were simultaneously in the same microcell. Thus, all these ephemerals are associated with users at risk of contagion. The server has to broadcast these ephemerals. Every smartphone can check if some of the received ephemerals is stored in the set of past ephemerals (recall that ephemerals are periodically changed) not older than the safety time window (i.e., 14 days, according to WHO). If this is the case, then the user is alerted about the risk. In the example above, if Alice is tested positive, the ephemerals of Alice of the last 14 days are notified to the server. Consequently, the ephemeral used by Bob when they came into contact, will be included in the list of ephemerals broadcast by the server. Therefore, Bob will be alerted by his smartphone. Regarding the infection reporting, we specify that the notification of past ephemerals by the infected user is done only if it is authorized by the health facility which tested the patient. This is done by using a blind-signature mechanism to avoid that the health facility can link the real identity of the patient with the transmitted ephemerals. Observe that the smartphone has to keep only the ephemerals of the last 14 days. Actually, some small additional information is kept by the smartphones to associate the alert, if any, with a level of risk. Therefore, no significant storage overhead is required to clients. Importantly, the fact that the ephemeral identifiers are never exchanged among smartphones is the basis of the improvements of our protocol in terms of privacy and security with respect to DP-3T/GAEN. This is clearly shown in the last part of the chapter, when security aspects are addressed.

6.4 The Proposed Protocol

In this section, we describe the proposed protocol, named *Zero Ephemeral Exchanging Privacy-Preserving Proximity Tracing* (ZE2-P3T, for short). As briefly anticipated in Section 6.2, the protocol relies on the presence of a sufficiently precise localization system utilizable with sensors on board of smartphones. Although this aspect is not within the scope of this work, it is important to highlight that the availability of such a system is not an abstraction. For example, the combination of GPS and PDR (Pedestrian Dead Reckoning) [101] allows for both indoor and outdoor localization. PDR is an algorithm that allows us to estimate the movement of pedestrians, using MEMS sensors, i.e., accelerometer, gyroscope, and magnetometer, on board of the smartphone. In particular, for indoor environments, there are different measurement techniques [159] such as *Angle of arrival (AOA)*, *Cell Identity (CI)*, *Time of Arrival (TOA)*, *and Signal Strength (RSSI)*, or the Earth's magnetic field. The usage of Earth's magnetic field is supported by results available in the literature like [57], which presents a system able to guarantee a maximum positioning error of less than 10 cm in an internal environment. To achieve greater accuracy, new technologies such as 5G and future developments, like 6G [35], can be used, also to obtain a continuous localization service between external and internal environments [107]. Positioning accuracy can be significantly increased by using the features offered by 5G [49], such as wider bandwidth (mmWave frequencies), data from a number of sensors and technologies (WiFi, GNSS, relative device-to-device positioning, inertial measures, etc.), and a dense infrastructure. Furthermore, in 5G, many new technologies have been proposed, such as massive Multiple Input Multiple Output (MIMO), millimeter Wave (mmWave) communication, ultra-dense network (UDN), and device-to-device (D2D) communication, which improve both communication performance but also positioning accuracy [104]. Moreover, the new millimeter-wave technology, being studied for 5G communication systems, will be able to provide centimeter-accurate internal localization in a robust way [179]. 6G technology, following the trend initiated by 5G systems, will continue to develop towards even higher frequency ranges, wider bandwidths, and massive antenna arrays. This will also allow for localization with a degree of accuracy at the level of the centimeter [35]. 6G technology, therefore, will not only provide ubiquitous communications but will also enable high-precision localization and high-resolution sensing services. Therefore, in the description of the protocol, we generically refer to a *localization system*, without considering a specific technology.

6.4.1 Protocol Setting

Also in this scenario, we use the tag-grid-based approach presented in Chapter 3, in Section 3.2. Let G be a large geographical area, for example, a country, in which the contacts among users have to be tracked. In our model, G contains several *microcells* c such that:

1. they cover all the area G;
2. if the distance between two points x and y is less than a threshold parameter d, then there exists a microcell which contains both x and y.

In our setting, the microcells are squares of side $2d$ organized as in Fig. 6.1. Referring to Figure 3.3, again in Chapter 3, in this case, we are considering the regions associated with the leaves of the tree. Therein, we use different colours to better highlight the different microcells (13 in total). It is easy to see that a point is always, simultaneously, within two different microcells and that two points at a distance less than d have a microcell in common. For example, in the figure, the point x is within the blue and green microcells while the point y is within the red and green microcells.

With each microcell c, we associate a point C called *centroid* corresponding to the center of the square. The set of all the centroids is public and each user, through the localization system, is able to identify the centroids associated with the two microcells where the user is located.

The actors involved in our protocol are:

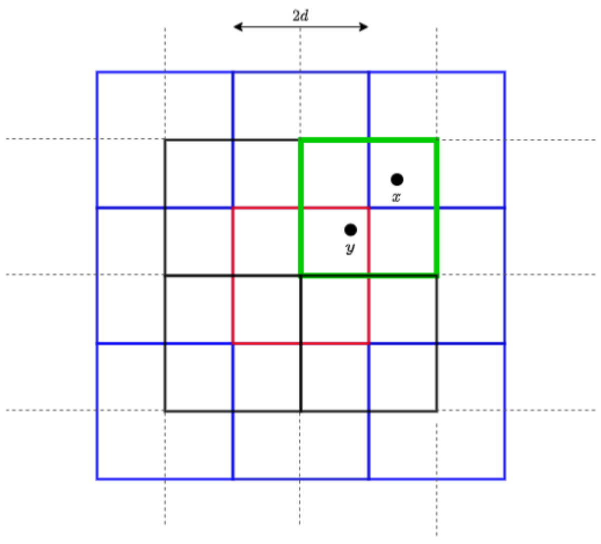


Fig. 6.1: Microcells organization.

- The *Users* moving within the territory. Each user owns a smartphone equipped with the localization system.
- The *Server* that receives the information sent by users' devices and stores the contacts in pseudonymous form (yet unlinkable).
- The *Health Facility* HF that performs the tests on the users to diagnose the disease and enables an infected user to send the information needed for infection reporting to the server.
- The *Telephone Service Provider* TSP which, periodically, for each zone P (covering a number of microcells), sends a different random R_P (called *salt*) to all the users in a *zone*. The zones cover the entire territory.

The role played by the first three actors is the same as DP-3T (even though they act in a different way). We provide some details about the TSP. The general operating principle is that presented in Chapter 3, obviously, in this chapter, it is adapted to the context dealt with. The role of TSP is to provide an unpredictable value to the users physically presents in a given zone. This value is used for salting the hash applied to the centroid, in such a way that the digest cannot be reversed by the server (to disclose the position).

We assume that every zone contains several microcells. As in real-life cellular systems the coverage overlap is at least 1 meter, we can argue that if two users are within the safety distance (of the magnitude of 1-2 meters, according to WHO), they receive at least one common salt if d is fixed to the same value. Note that, due to the overlapping of two zones P_1 and P_2, the same user can receive two salts R_{P_1} and R_{P_2}.

R_P is rounded by TSP with a certain frequency, which is a parameter of the system (we assume that this can be set in accordance with the average permanence time of a user in a zone).

For simplicity, we assume that this service is provided by a unique TSP (to avoid to treat the problem of coordination of multiple TSP in overlapping cells) and that the roaming mechanism can be enabled to ensure maximum coverage.

Before going into details, we provide some security assumptions.

- TSP, Server, and HF are assumed to be TTPs (Trusted Third Parties);
- TSP is assumed to be independent;
- HF and Server cannot be considered independent because they could be part of the same National Health System.

In Section 6.7, concerning the adversary model, we relax the above assumptions by giving to the above parties also the role of adversary.

We distinguish five logical phases in our protocol.

6.4.2 User-side Activity

Each τ seconds, each user U detects, through the localization system, the two centroids C_1 and C_2 of the microcells in which they are located. Moreover, they detect one or two salts provided by the TSP depending on whether the user is located just in a single zone or in the overlapping between two zones. We consider the more general case of two salts R_{P_1} and R_{P_2}.

For each pair of centroid-salt, the user builds a tuple of the form $T = \langle Eph, H, (\rho, \theta), t\rangle$. Eph is said *ephemeral identifier*, H, when the context is clear, is called simply *digest*, (ρ, θ) are called *coordinates*, and t is called *timestamp*.

We show in detail how these tuples are built in the most complex case of four tuples:

1. $T_1 = \langle Eph_1, H_{11}, (\rho_1, \theta_1), t\rangle$,
2. $T_2 = \langle Eph_2, H_{21}, (\rho_2, \theta_2), t\rangle$,
3. $T_3 = \langle Eph_3, H_{12}, (\rho_1, \theta_1), t\rangle$,
4. $T_4 = \langle Eph_4, H_{22}, (\rho_2, \theta_2), t\rangle$.

where Eph_i $(1 \leq i \leq 4)$ is a random value, $H_{ij} = h(C_i \| R_{P_j})$ $(i, j \in \{1, 2\})$ denotes the application of a cryptographic hash function on the concatenation between a centroid and a salt, (ρ_i, θ_i) (where $i \in \{1, 2\}$) are the relative polar coordinates of U with respect to the centroid C_i, and t is the current UNIX timestamp (the same for all the four tuples).

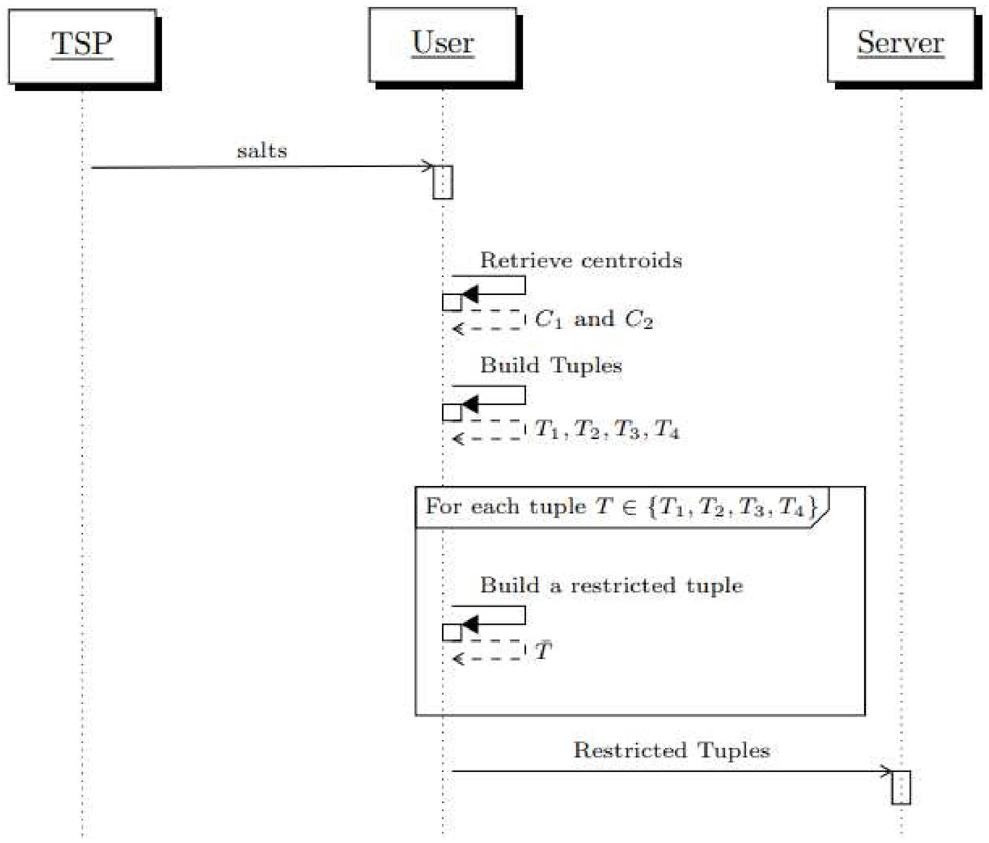


Fig. 6.2: Sequence diagram of the user-side activity.

These four tuples are stored locally by U for a given time interval, that we call *retention time*, (e.g., 14 days, according to the original recommendation given by WHO). Observe that in the case that a single salt R_{P_1} is retrieved by U_x, just the first two tuples are generated.

As discussed in Section 6.7, the role of the salt is to avoid dictionary attacks performed by the server with the aim to identify the centroids (and then the location) of the users.

Now, for each tuple $T = \langle Eph, H, (\rho, \theta), t\rangle$, a *restricted* tuple $\bar{T} = \langle Eph, H, t\rangle$ is sent to the server, accordingly the restricted tuples do not include the relative coordinates of the users. Their function is to improve the accuracy of the risk computation as discussed in Section 6.4.6 that is performed client-side and only in case of need.

The sequence of actions performed in this phase is reported in the sequence diagram of Figure 6.2.

6.4.3 Server-side Activity

We assume a time slot mechanism is enabled. Specifically, the time is partitioned in time slots $\tau_k = [t_k, t_{k+1}[$, where t_k and t_{k+1} are UNIX timestamps such that $t_{k+1} - t_k = \tau$. The size of the time slot is a constant value equal to the sending period of the users.

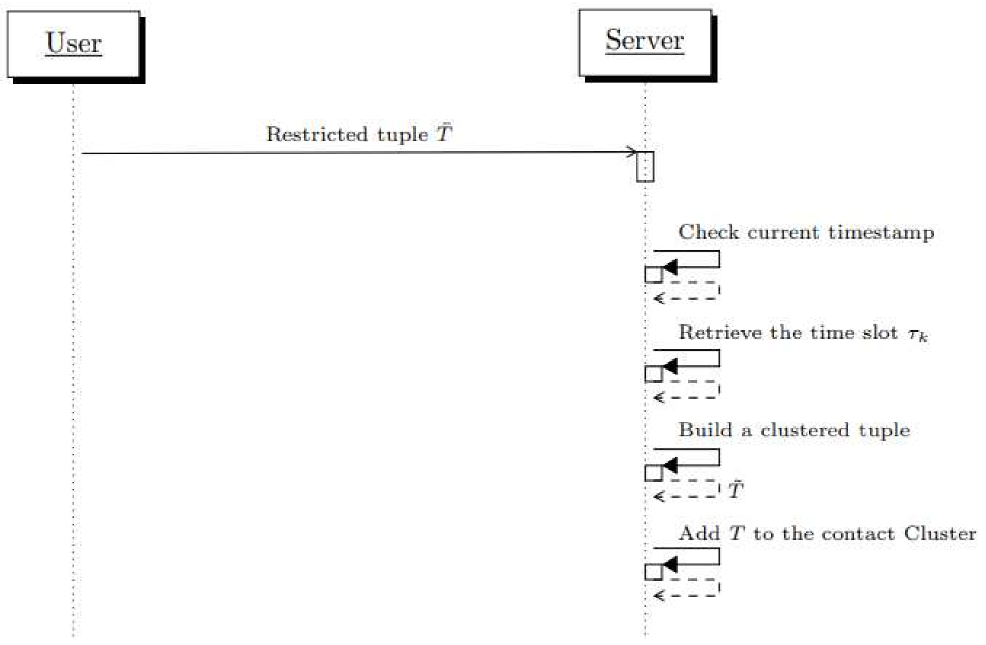

User
Server
Restricted tuple $\bar{T}$
Check current timestamp
Retrieve the time slot τ_k
Build a clustered tuple
$\tilde{T}$
Add T to the contact Cluster

Fig. 6.3: Sequence diagram of the server-side activity.

For each received restricted tuple $\bar{T} = \langle Eph, H, t \rangle$, the server verifies that the current UNIX timestamp is not too greater than t and retrieves the time slot τ_k such that $t \in \tau_k$. Then, it builds a *clustered tuple* $\tilde{T} = \langle Eph, H, \tau_k \rangle$. The term clustered derives from the fact that the server maintains, for each time slot τ_k, a list of *contact clusters*. A *contact cluster* is a set of clustered tuples with the same time slot and same digest. A contact cluster represents a set of users that shares a centroid in the same time slot, i.e., users experienced unsafe contacts.

To avoid the inflationary growth of the server-side database size, the server maintains only the information of the last X days (with X suitably greater than the retention time).

The sequence of actions performed in this phase is reported in the sequence diagram of Figure 6.3.

6.4.4 Infection Reporting

Suppose the user U is tested positive for the infection in the health facility HF. The infection reporting requires that U sends the server the non-restricted tuples generated as described in Section 6.4.2 in the last X days, where X is the retention time. We denote by $\mathcal{T}$ the set of such tuples.

In order to avoid fake positive reports, U needs authorization by HF. We rely on a 1024 bits RSA blind signature scheme. As discussed in Section 6.7, the blind signature also avoids that, even though the server colludes with HF, it is not able to link all the provided tuples to the real identity of U. The procedure is the following.

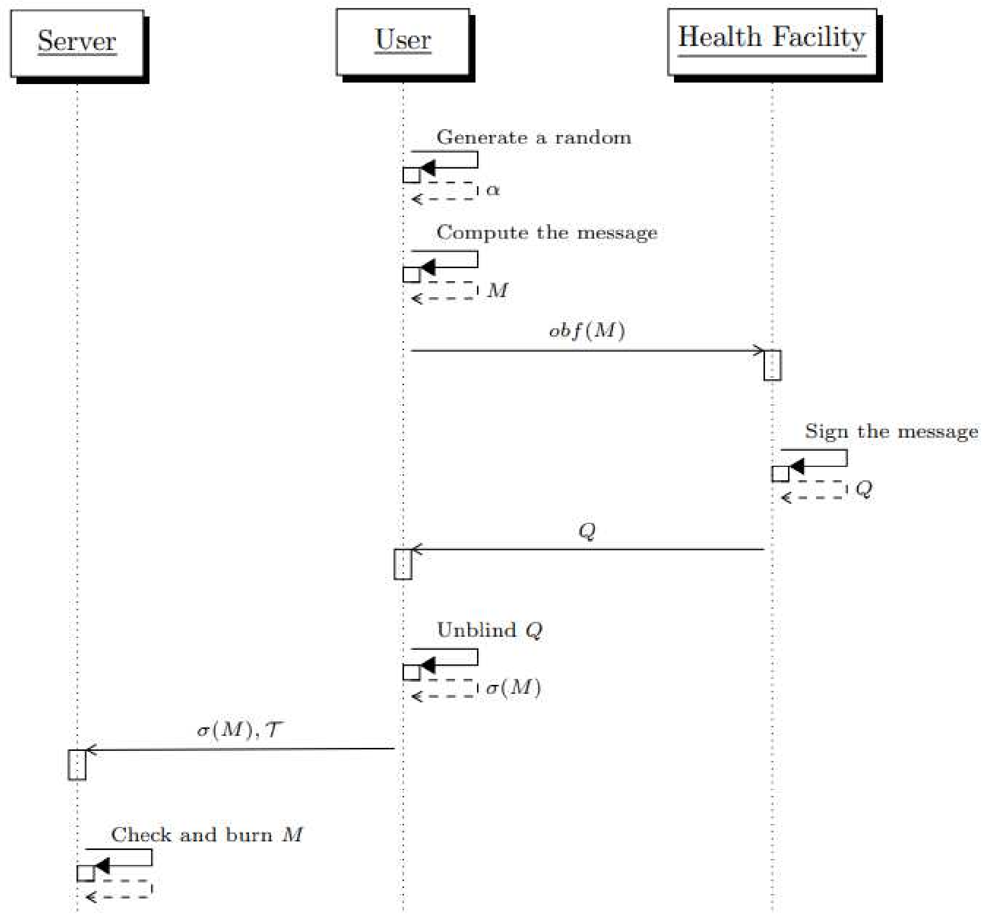


Fig. 6.4: Sequence diagram of the infection-reporting phase.

First, U generates a random α of 1024–256 = 768 bits and obtains $M = \alpha \| h(\alpha)$, where h denotes the application of a cryptographic hash function with digests of 256 bits (e.g., SHA-256). At this point, U contacts HF to obtain the RSA blind signature on M. This is done by sending HF an obfuscated message $obf(M)$ unlikable to M. Let denote by Q the message with blind signature produced by HF. U unblinds Q and obtains the signature of HF $\sigma(M)$ on M. Finally, at a later moment, U sends $\sigma(M)$ and all the non-restricted tuples they locally stored in the server.

The server verifies the signature and checks that $M = \alpha \| h(\alpha)$. To avoid replay attacks, the server burns the random M, so that it cannot be used anymore.

The infection-reporting procedure is summarized in the sequence diagram of Figure 6.4.

6.4.5 Contact Detection

After receiving a set of tuples $\mathcal{T}$ by an infected user, the server performs the following actions. For each tuple $T \in \mathcal{T}$ with $T = \langle Eph, H, (\rho, \theta), t \rangle$, the server retrieves the time slot τ_k such that $t \in \tau_k$. Then, it retrieves the contact cluster $\mathcal{C}$ associated with

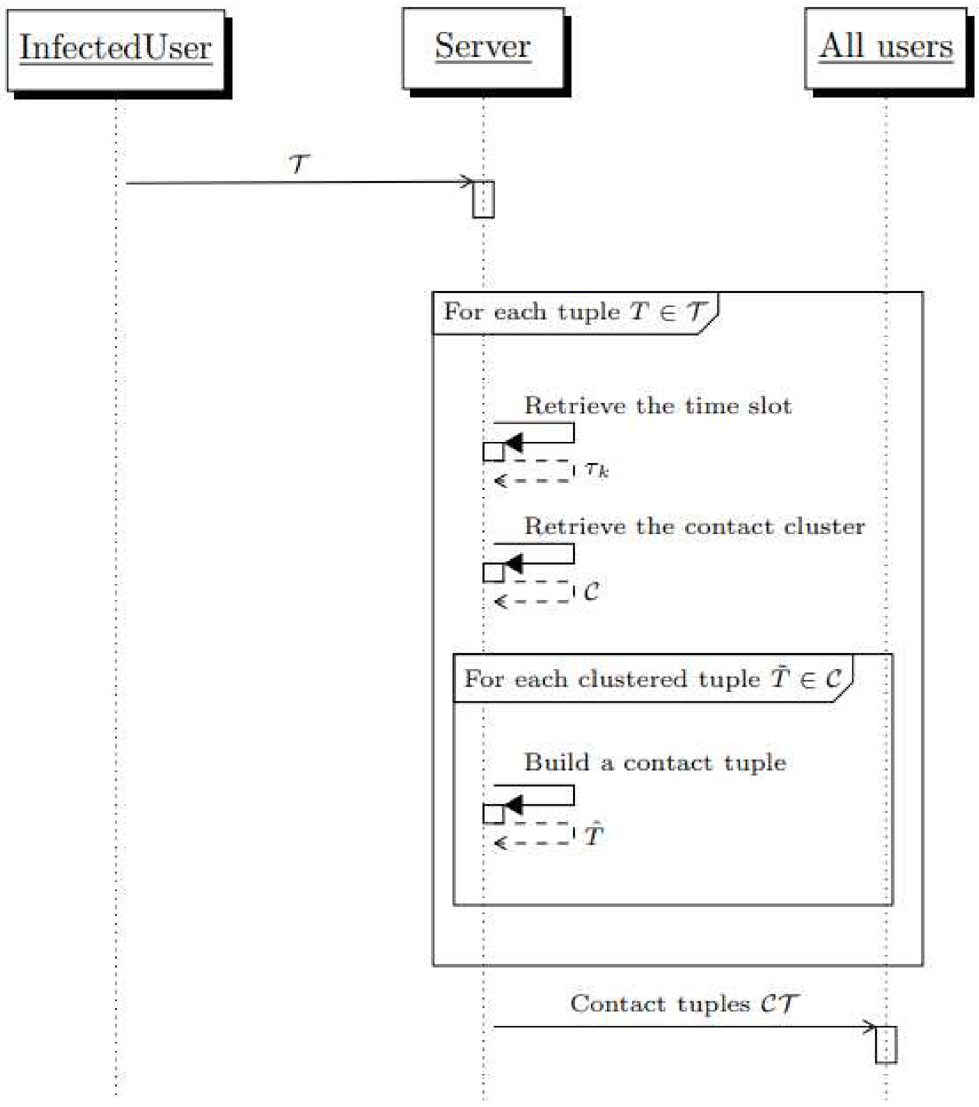

InfectedUser
Server
All users
$\mathcal{T}$
For each tuple $T \in \mathcal{T}$
Retrieve the time slot
τ_k
Retrieve the contact cluster
$\mathcal{C}$
For each clustered tuple $\tilde{T} \in \mathcal{C}$
Build a contact tuple
$\hat{T}$
Contact tuples $\mathcal{CT}$

Fig. 6.5: Sequence diagram of the contact-detection phase.

the digest H. If two distinct tuples fall within the same time slot, the server randomly chooses one of them and skips the other. This is done to avoid that a malicious user can fictitiously increase the number of ephemeral identities corresponding with that infection reporting. For each clustered tuple $\tilde{T} = \langle Eph', H, \tau_k \rangle \in \mathcal{C}$ with $Eph' \neq Eph$, the server builds a *contact tuple* $\hat{T} = \langle Eph', H, \tau_k, (\rho, \theta) \rangle$ that includes the relative coordinates of the infected user.

The set of all the contact tuples, is included into a single file, called say $\mathcal{CT}$ (*contact file*) collecting all the information about the infection reports regarding other infected users. The file is transmitted in broadcast periodically, and then erased in order to collect the new infection reports.

The contact-detection phase is summarized in the sequence diagram of Figure 6.5.

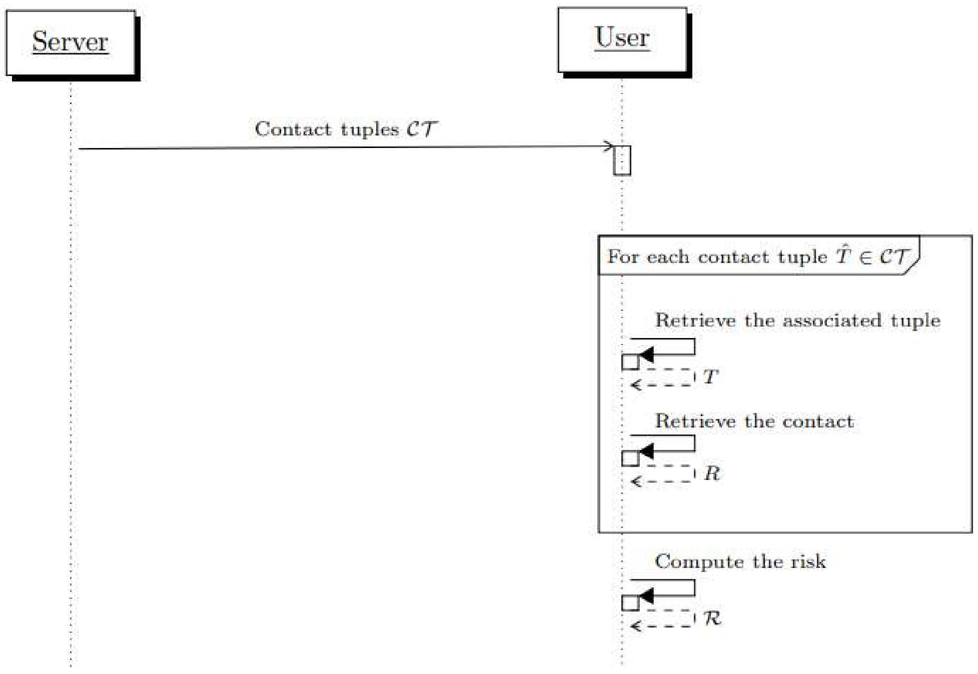

Server
User
Contact tuples $\mathcal{CT}$
For each contact tuple $\hat{T} \in \mathcal{CT}$
Retrieve the associated tuple
T
Retrieve the contact
R
Compute the risk
$\mathcal{R}$

Fig. 6.6: Sequence diagram of the risk-computation activity.

6.4.6 Risk Computation

After receiving the contact file $\mathcal{CT}$, each user U_y performs the following actions. For each contact tuple $\hat{T} = \langle Eph', H, \tau_k, (\rho, \theta)\rangle \in \mathcal{CT}$, U_y retrieves, if any, a tuple $T = \langle Eph', H, (\rho', \theta'), t\rangle$, i.e., a tuple with the same ephemeral identifier (and the same digest H). Then, it generates the pair $R = (d, \tau_k)$, where $d = \sqrt{\rho^2 + \rho'^2 - 2\rho\rho' cos(\theta - \theta')}$. R represents a contact between U_y and an infected user in the time slot τ_k at a distance d. We denote by $\mathcal{R}$, the set of all the pairs computed by U_y The risk value r for U_y is a function of $\mathcal{R}$.

We do not focus on the function r for the computation of the risk level since it depends on several medical factors. We can say that the function increases as the cardinality of $\mathcal{R}$ increases and it decreases as the distances between users increase. Moreover, consecutive time slots represent a prolonged contact, then a higher risk.

We just remark that all the information typically used to evaluate the risk in digital contact tracing solutions is available also in our model.

The sequence of actions performed in this phase is reported in the sequence diagram of Figure 6.6.

6.5 Implementation and Experiments

In this section, we describe the experimental environment, the prototype we developed to validate our proposal, and the experimental procedure we followed.

6.5.1 Experimental Environment

The experiments conducted are aimed at analyzing the performance of our protocol. To conduct our experiments, we used a prototype implementing our solution whose description is given in Section 6.5.2.

To have realistic results, we leveraged the Thomas Brinkhoff data generator [37] to simulate n users moving in a city. As city we choose the default city of the simulator, which is Oldenburg (Germany). The Thomas Brinkhoff data generator uses a discrete time model to simulate different moving patterns of objects (belonging to different classes) in real networks. To test the growth of clients we developed a java multi-thread application in which each thread implements the part of the (android) client-side prototype responsible for the communication with the server.

Our experiments were performed by employing a personal computer equipped with 1.8 GHz Intel i7-8850 CPU and 16 GB of RAM.

Before discussing the experimental procedure, in the next section, we describe how we implemented the prototype.

6.5.2 Implementation

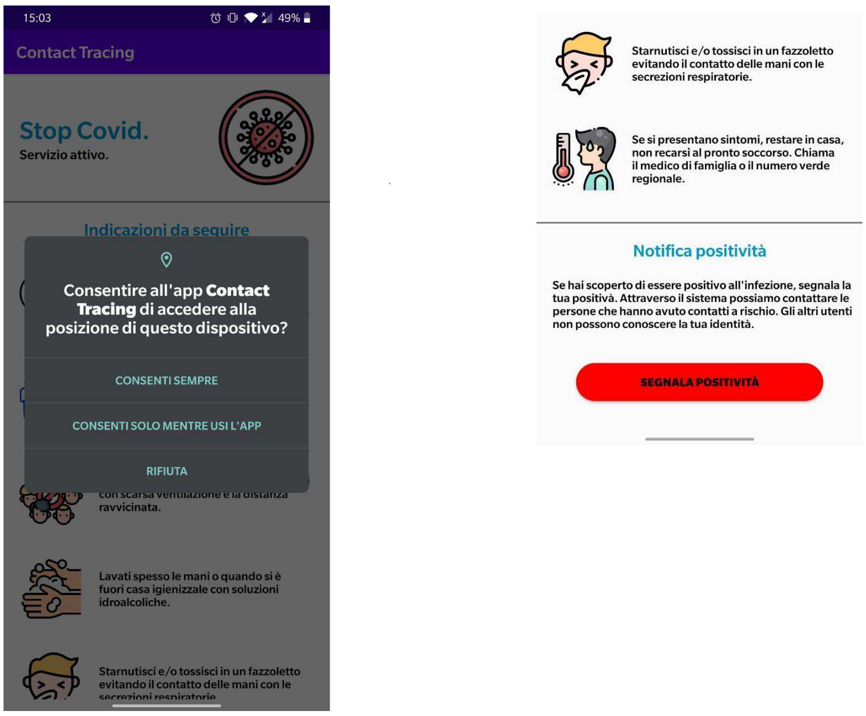
In this section, we describe the prototype we developed, which implements the main functionalities of ZE2-P3T. The source code of this implementation is available at `https://github.com/vincenzodeangelisrc/ZE2-P3T`. Compared with the previously presented protocol, the prototype includes some optimizations client-side (see below for detail), that should be disabled in order to refer to the security features shown in Section 6.7.

The implemented modules are three: (1) *Client*, (2) *Health Facility*, and (3) *Server*.

Client Module

The Client module is a mobile Android application written in Java. We report two screenshots of the Welcome page (written in Italian) in Figure 6.7.

The application includes two Activities. The `Welcome Activity` has only graphical functions and shows some alerting messages. The core of the module is the `Main Activity` that implements the ZE2-P3T protocol (i.e., generation of the ephemeral IDs, retrieval of the centroids, and sending information to the server). The retrieval of the salt from the TSP is simulated. A full implementation of this feature would require the collaboration of a telephone service provider to set the technological detail regarding the transmission of the salt. We plan to implement it as future work, by involving in our experimentation also industrial partners (this is the reason why, at moment, the interface of the application is developed in Italian). The application


15:03 49%

Contact Tracing

Stop Covid.
Servizio attivo.

Indicazioni da seguire

Consentire all'app **Contact Tracing** di accedere alla posizione di questo dispositivo?

CONSENTI SEMPRE

CONSENTI SOLO MENTRE USI L'APP

RIFIUTA

con scarsa ventilazione e la distanza ravvicinata.

Lavati spesso le mani o quando si è fuori casa igienizzale con soluzioni idroalcoliche.

Starnutisci e/o tossisci in un fazzoletto evitando il contatto delle mani con le secrezioni respiratorie

Starnutisci e/o tossisci in un fazzoletto evitando il contatto delle mani con le secrezioni respiratorie.

Se si presentano sintomi, restare in casa, non recarsi al pronto soccorso. Chiama il medico di famiglia o il numero verde regionale.

Notifica positività

Se hai scoperto di essere positivo all'infezione, segnala la tua positivà. Attraverso il sistema possiamo contattare le persone che hanno avuto contatti a rischio. Gli altri utenti non possono conoscere la tua identità.

SEGNALA POSITIVITÀ

Fig. 6.7: Screenshots of the Welcome page.

also includes a `Foreground Service` that allows the users to note that some operations are performed by the app and consume resources of the system.

The `Notification Exposure` component implements the client-side activity of the infection-reporting phase described in Section 6.4.4. The generation of the random α of 768 bits is performed through the class `SecureRandom` and the selected hash algorithm is SHA-256.

As mentioned earlier, the code of the Client module includes some optional optimization features, differentiating the implementation from the protocol presented in Section 6.4. These features are based on BLE and have the aim to reduce the effort performed client-side and server-side. In particular, the idea is to use BLE to detect the proximity of the users and to send data to the server only in this case (instead of the periodical sending). Even though some information are exchanged through BLE between users, they do not include the ephemeral IDs (thus, preserving, in principle, the security features offered by ZE2-P3T). However, no formal analysis is conducted about this aspect (planned as a possible extension of the protocol and future work). Therefore, in the current state, the BLE component should be disabled to have security guarantees. Certainly, enabling the BLE interface exposes our application to attacks such as those reported in [63].

Health Facility Module

The HF module is a web application implemented in Java by relying on the Servlet technology. The servlet `BlindSignature` simply receives the message to sign from the client module, invokes the method `calculateSignatureOfMessage`, and returns the signed message to the client module.

The core of the HF module is the `Health Facility` class that contains the method `calculateSignatureOfMessage`. It implements the standard RSA blind signature. For the final implementation of the system, we plan to exploit tested crytographic libraries.

Server Module

Also the Server module is a web application implemented through the Servlet technology.

It includes two servlets (`RestrictedTuplesReport` and `InfectionReporting`) and six classes (`Tuple`, `RestrictedTuple`, `Clustered Tuple`, `ContactTuple`,`Key`, and `ContactClusters`). The first four classes, as the names reveal, represent the four types of tuples present in our protocol. The class `Key` is an accessory class representing the key of a HashMap contained in the `ContactClusters` class. Such a key is composed of a time slot and a digest. The HashMap is of the type `ConcurrentHashMap` that is thread-safe, so that multiple parallel requests coming from the clients can be managed. This HashMap associates each `Key` with a Set of `ContactTuple` representing a contact cluster as defined in Section 6.4.3.

Also the Set of Contact Tuples is obtained as a particular implementation of a `ConcurrentHashMap` to be thread-safe.

The core of this module is represented by the two servlets. The servlet `RestrictedTuplesReport` implements the server-side activity of Section 6.4.3. It is called by the users to provide the server with the restricted tuples sent periodically. From these tuples, the servlet retrieves the contact tuples and fills the HashMap of the class `contactClusters`. In addition, periodically, at each time slot, the HashMap is emptied and its content is stored in a back-end database. In the provided implementation, we rely on MySQL 8.0. As optimization, if a Contact Cluster contains just a restricted tuple (i.e., there is a single user in a microcell in a given time slot), it is not stored in the database.

The servlet `InfectionReporting` implements the server-side activity of Section 6.4.4 and Section 6.4.5. This servlet receives the non-restricted tuples by the positive users and detects the contacts by the database. Then all the contacts (along with those produced by other infected users) are stored in the contact file that will be downloaded periodically by the users.

6.5.3 Experimental Procedure

In this section, we describe the experimental procedure we followed to test our protocol. The experiments aimed at validating the feasibility of our proposal. Observe that, client-side, the smartphone has to perform very cheap autonomous operations. Indeed, the smartphone, during the standard client-side activity, is required to perform:

1. detection of (at most) two centroids and salts;
2. computation of (at most) 4 digests;
3. sending of (at most) 4 restricted tuples to the server.

Also the number of operations performed during the infection reporting phase is small. It generates a random number, computes an obfuscated message for the blind signature, unblinds the message returned by the health facility, and uploads some tuples to the server. Regarding the contact detection phase, after downloading the contact file, the client has to find its ephemerals in this file to find.

Obviously, considering the standard computational capabilities of current smartphones, the above operations are definitely feasible.

A similar consideration can be done for the operations performed by HF.

Therefore, the experiments focused on the tasks performed server-side. Indeed, being our approach centralized, the server-side component of the computation could represent, in principle, an issue of the solution, also because the application should be designed for a huge number of clients. Through the performed experiments, we show that, even with limited computational resources (i.e., a standard PC), the computation can be performed also for a high number of users. On the other hand, for how they are designed, the experiments can be also view as an operational method to size the server-side computational and storage resources in function of the operation conditions.

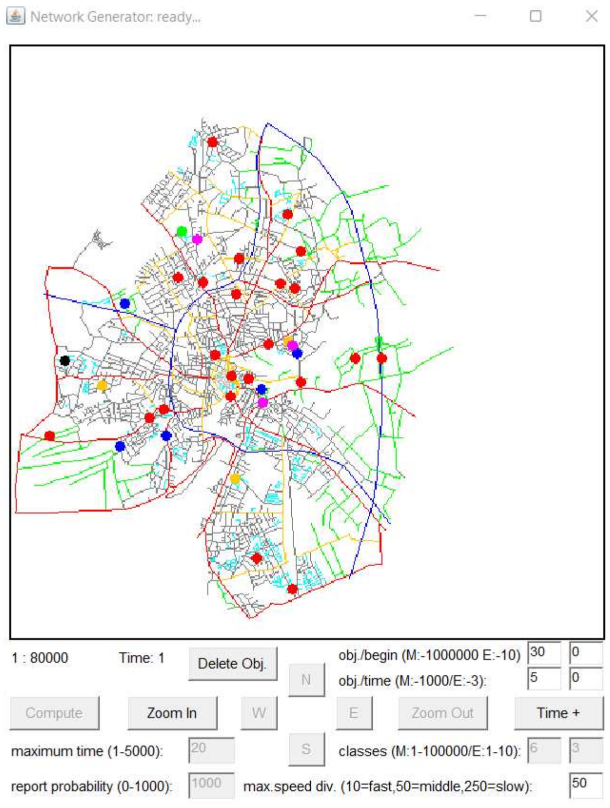
In our experiments, we considered an observation window of 14 days. This value represents the retention time (see Section 6.4) after which the data produced by users are discarded. Then, we set the time of the time slots τ (sending period of the users) equal to 5 minutes, the same as DP-3T/GAEN. The size of the microcells in which the territory is partitioned is $2x2m^2$.

Two types of experiments have been performed.

In the first experiment, we analyzed the server-side computation during the standard client-side and server-side activities.

In particular, we simulated a number n of users, moving in the city of Oldengurb through the Thomas Brinkhoff generator, who, at each time slot, transmit their restricted tuples to the server.

A graphical representation of the city and users is depicted in Figure 6.8.



1 : 80000 Time: 1

obj./begin (M:-1000000 E:-10) 30 0

obj./time (M:-1000/E:-3): 5 0

maximum time (1-5000): 20

classes (M:1-100000/E:1-10): 6 3

report probability (0-1000): 1000

max.speed div. (10=fast,50=middle,250=slow): 50

Fig. 6.8: Users distribution in the Oldengurb city.

Due to the limited computation power of the employed PC, we considered a number of users n ranging from 100000 to 600000. However, we show in Section 6.6, both analytically and experimentally, that the processing time increases linearly with n. Then, it suffices to employ a multi-core CPU to easily manage some millions of users.

Through the multi-thread application, we generated each 5 minutes (size of the time slot), the parallel requests coming from the n users and measured the processing time server-side.

In the second experiment, we analyzed the server-side computation time during the infection reporting phase. Specifically, we varied the number I of infection reporting requests performed by the users positive to COVID-19 in a given interval of time T and measured the processing time server-side. For uniformity, we consider I as the number of infection reporting requests performed in 5 minutes.

Moreover, we evaluate the quantity of bytes required to maintain the data in: (1) central memory of the server, (2) secondary memory of the server, (3) the contact file.

(3) is the most critical factor, since this file is downloaded periodically (e.g., daily) by the users. Then, we proposed an improvement which is applicable if the exact polar coordinates of the positive users are not stored in the file. The idea is to store an approximate version of these coordinates. In particular, we split each microcell in t subcells and include in the contact file just the identifiers of the subcells in which the positive users are located.

This way, the ephemeral identifiers of the contact tuples are concatenated with the identifiers of the subcells and can be efficiently stored in a Bloom filter [160]. A Bloom filter is a probabilistic data structure used to test whether an element is a member of a set. The probabilistic nature of the data structure regards only false positives, which are possible. Instead, false negatives are not possible. This property makes Bloom filters suitable for our application. Indeed, it works in favor of safety. However, to minimize false positives, we set the false positive probability to a very low value, namely 10^{-6}. This, in practice, means that there are no false positives. With Bloom filters, we drastically reduce the size of the contact file.

Client-side, when the contact file is downloaded, the smartphone can check if one of its ephemeral identifiers is in the contact file by testing all the t subcell identifiers. In our setting, we choose $t = 16$, i.e, we split the microcell into 4x4 subcells each of side $50cm$. This value allows us not to pay a relevant price client-side (16 searches in the Bloom filter per ephemeral). Once the subcell is retrieved, the client computes the distance between the locally stored coordinates and the center of the subcell. Observe that given the small size of the subcell, the computed distance is quite accurate.

Another point to take into consideration is whether the use of the Bloom filter introduces a server-side computational overhead during the infection reporting phase. Then, in the experimental procedure, we repeated the second experiment by including the Bloom filter. As we will see in the next section, the adoption of Boom filters does not introduce drawbacks server-side.

6.6 Results

This section has a dual purpose. First, we perform a (space and time) cost analysis of the server-side tasks considered in Section 6.5.3. The cost analysis allows us to foresee the scalability of our solution, since, basically, we obtain costs that grow linearly with the size of the input. Through experiments, we give the exact costs in the considered experimental setting, which confirm the above prediction.

6.6.1 Cost Analysis

We start by considering the time required server-side to process the requests coming from the clients during the client-side and server-side activities (first experiment of Section 6.5.3).

Each τ seconds, each client provides the server with at most 4 restricted tuples. In the proposed implementation, each tuple is inserted in the HashMap with constant cost. By considering n users, there are (at most) $4n$ insertions per time slot. Finally, the content of the HashMap is stored in the database. We set a Hash index on the `Digest` attribute of the restricted tuple, so that both the insertion and the search of a tuple require constant cost. Then, the time to store the HashMap in the secondary memory is proportional to the size of the HashMap (i.e., at most $4n$). We can conclude that the computational cost required server-side per time slot τ is $O(n)$.

Clearly, as we will discuss in the next section, the transfer from the HashMap to the database is the dominant operation. Furthermore, observe that this linear growth occurs until the required time is less than the time slot size (i.e., 5 minutes). Indeed, above this threshold, a new burst of requests reaches the server before it ends to process the previous burst, and then, the performance degrades. Also this point is discussed in the next section.

Consider now the processing time required during the infection reporting phase.

When an infection reporting request reaches the server, it performs a query on the database to find the contact tuples associated with a given digest and time slot.

To improve the performance, we included the time slot in the digest itself, so that the search can be done efficiently by digest. Being the Hash index built on this attribute, the search cost of a single digest is constant.

We recall that an infected user reports the tuples which they generate in the previous 14 days. This number of tuples is constant and independent of the total number of users.

After performing the search, the query result is stored in a file (or in the Bloom filter). Even though, in principle, this cost depends on the size of the query result, we can say that this number is very small (almost constant) compared with the total number of users. Indeed, it depends on the number of users simultaneously present in a microcell of $2x2m^2$, which is obviously very small.

Then, the search cost of a single infection reporting is $O(1)$. Then, given I infection reporting requests per time slot, time required to the server to solve them is

$O(I)$. We observe that, also by enabling the Bloom filter, the insertion cost of a single ephemeral is constant. Then, asymptotically, the cost remains $O(I)$. However, we experimented a small difference by comparing the actual costs of our solution with and without the Bloom filter, respectively.

Consider now the space required to store the data of our protocol. To be general, we denote by X the observation time (i.e, 14 days) in seconds, and by τ the size of the time slot in seconds. Moreover, we denote by k the average number of users simultaneously present in a microcell.

First, in the central memory the server loads the Hash Table, which is emptied at each time slot τ, and the contact file, which we assume is downloaded (and then emptied) daily by the users.

Regarding the size of Hash Table, consider that, each τ seconds, at most $4 \cdot n$ restricted tuples have to be inserted. If we denote by x the size (in bytes) of a single restricted tuple, the maximum space required for the Hash Table is $4 \cdot n \cdot x$ bytes.

Consider now the storage space required in the secondary memory. Since it contains the tuples stored in the last 14 days (X seconds) by the users, this space will be $4 \cdot n \cdot \frac{X}{\tau} \cdot x$. Actually, the tuples generated by users who do not share any microcell with other users, are not stored. Then, the above analysis represents the worst case.

Finally, consider the contact file. The size of this file depends on the number of contacts of the users in the last 14 days. We start by considering that a single positive user reports all the tuples they generate in X seconds i.e., $4 \cdot \frac{X}{\tau}$. For each tuple, other $k-1$ (clustered) tuples are retrieved by the secondary memory.

Then, the contact file will contain $4 \cdot \frac{X}{\tau} \cdot (k-1)$ contact tuples per positive user.

If we denote by y the size of a contact tuple and by i the average number of positive users (per day), we have that the size of the contact file will be: $4 \cdot \frac{X}{\tau} \cdot (k-1) \cdot i \cdot y$.

If we adopt the Bloom filter, the size of the file can be computed through the standard formulas. In particular, given the (maximum) number of elements to be included in the filter and the probability of false positives, we can obtain the size in bytes of the filter (and the optimal number of hash functions to set).

Specifically, given d elements to include in the filter and a false positive probability p, the size of the filter will be [160]: $-\frac{d \cdot ln(p)}{(ln2)^2}$.

In our setting, $d = 4 \cdot \frac{X}{\tau} \cdot (k-1) \cdot i$.

As we will see in the next section, in practice, the size of the contact file is drastically reduced by the use of the Bloom filter.

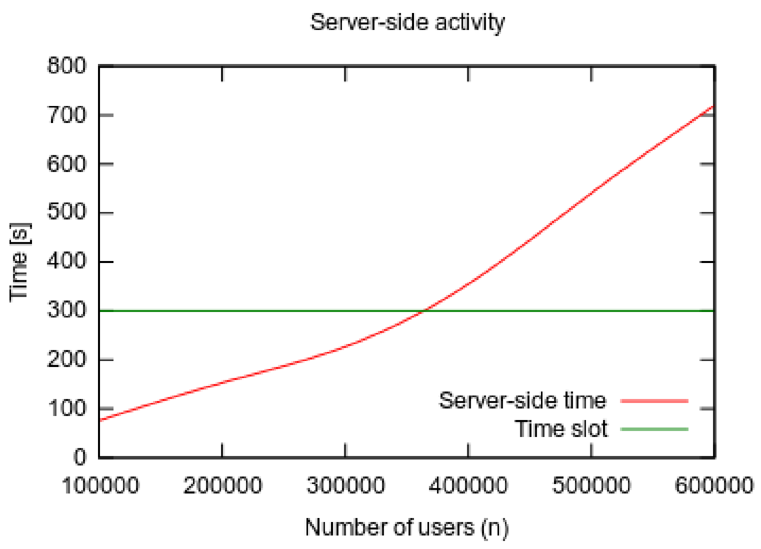


Fig. 6.9: Computational time required for the server-side activity.

6.6.2 Experimental Results

Now, we can describe the results we obtained through our experiments. The plot in Figure 6.9 shows the computational time required to the server to process the requests coming from *n* users in a time slot.

We observe that the time increases linearly with the number of users. This is coherently with the previous analysis. This happens until the time reaches the time slot size (green line in the plot). After this threshold, non-linear effects are present, due to the fact that a new burst of tuples reaches the server before it processes entirely the previous burst.

During the experiment, we observed that the transfer time of the Hash table from the central memory to the secondary memory represents about the 90% of the total time required to the server during this phase. The other 10% of the time is spent to process the incoming requests and store the tuples in the Hash Table.

This plot also shows that the personal computer used for the experiments, despite its standard capabilities, is able to manage up to 300000 users. Then, by considering the linear increase, a dedicated platform (possibly in the cloud) can easily manage millions of users. Actually, the above experiments, thanks to threshold analysis, provide a method to size the resources server-side.

Consider now the infection reporting phase whose results are reported in the plots of Figure 6.10.

Therein, we show the computational time required to the server during an infection reporting when we enable the Bloom filter and when we disable it.

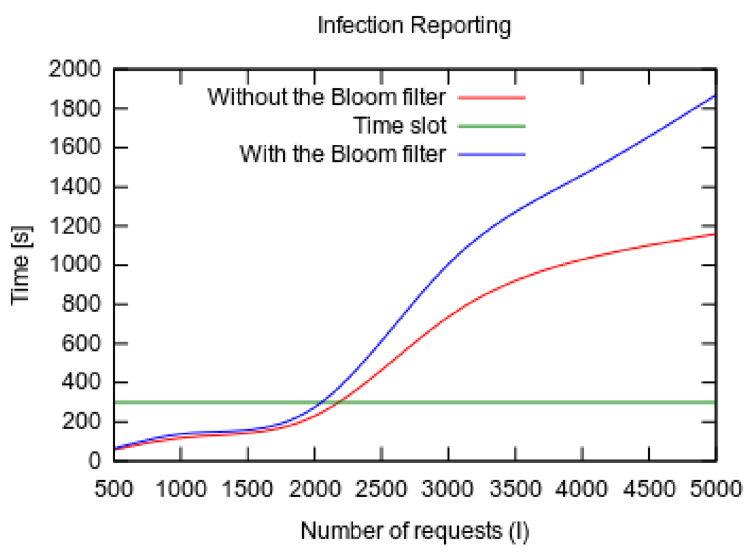


Fig. 6.10: Computational time required during the infection reporting phase.

The server computational time is reported as a function of the number of infection reporting requests I arriving in a time slot. In accordance with the cost analysis, we have that until the computational capacity of the employed PC is not saturated, the growth of the time is linear in I with and without the Bloom filter.

The computer used for experiments is able to manage up to 2000 requests per time slot. Observe that, the introduction of the Bloom filter (blue line) has a minimal impact on the performance until this value is reached.

We want to highlight that 2000 requests per time slot correspond to 576000 infected users per day. By relying on the aggregate data on the pandemic available at `https://github.com/CSSEGISandData/COVID-19` [59], we observe that in the peak period (January 2022), by considering just the countries with the most daily cases (in relation to the population), we have a daily rate of positively of 0.2%. This means that only a personal computer is able to manage about 288 millions of users. Thus, the infection reporting phase is definitively not critical.

We conclude by estimating the actual space required by our solution.

Considering the Hash Table, since each restricted tuple includes a 32-bytes digest, an 8-bytes ephemeral, and a 16-bytes time slot, the size of each restricted tuple is x=56 bytes.

For $n = 10$ millions of users, the Hash Table requires a space of $4 \cdot n \cdot x = 2.24$ GB that perfectly fits in the central memory of a standard personal computer. Clearly, we have to add the size of the contact file that resides in the central memory too.

	Without the Bloom Filter		With the Bloom Filter	
Memory	**1 million**	**10 millions**	**1 million**	**10 millions**
Central	486 MB	4.86GB	281 MB	2.81 GB
Secondary	903GB	9.03TB	903GB	9.03TB
Contact File	262 MB	2.62 GB	57,51MB	575,1 MB

Table 6.1: Space cost analysis.

Regarding the storage required in the secondary memory, it is the same as the central memory multiplied by $\frac{X}{\tau}$ =4032. This corresponds, for n = 10 millions, to 9.03 TB (again, a feasible value for standard storage devices).

The main bottleneck is represented by the size of the contact file. Indeed, the contact file should be downloaded periodically by the users. If we do not use the Bloom filter (to include the exact polar coordinates of the infected users), we recall that the size of the contact file is $4 \cdot \frac{X}{\tau} \cdot (k-1) \cdot i \cdot y$.

We set $k = 1.5$ users per microcells simultaneously (it means that for half of the time a user is alone in a microcell and for the other half the user is with another user). Then, given $i = 0.002 \cdot n$ and $y = 16$ bytes (we include in the contact file just the ephemeral ID of 8 bytes and the two coordinates of 4 bytes), we have that for 10 millions of users, the size of the contact file will be of 2.62 GB. Even though it can be stored in the central memory, it is unrealistic that a user downloads such a size each day.

Then, without the adoption of the Bloom filter, a realistic number of users that can be supported is 1 million leading to a size of the contact file of 262 MB.

If we introduce the Bloom filter, we reduce this size. In particular, given a false positive report probability of 10^{-6}, we obtain that the size of the contact file for n = 1 million is 57,51 MB and the size of the contact file for n =10 millions is 575,1 MB.

These results are summarized in the table reported in Table 6.1.

6.7 Security Analysis

In this section, we provide a security (comparative) analysis of ZE2-P3T and DP-3T protocols. In Section 6.3, we explained the protocol DP-3T and its designs, namely Low-Cost and Unlinkable. We start by introducing the following assumptions.

A0 *In the adversary model, TTPs can play as attackers.*

A1 *In the adversary model, two independent TTPs cannot collude.*

A2 *A sybil attack [22] proliferating fake users even with a passive role cannot require any action from the side of any non-attacker TTP.*

Note that Assumption **A1** implies that TSP and server (in ZE2-P3T) cannot collude. Indeed, they are independent TTPs. We emphasize the fact that, to be really applicable Assumption **A1**, the selection of the two parties should guarantee truly mutual independence. Furthermore, by considering that we can expect that the server is managed by a government institution, it is not realistic that a government colludes with the major telephone company of the country without risking that this attempt is discovered and results in dramatic consequences for the public opinion. To further prove that Assumption **A1** conforms to the facts, observe that, while assuming that a single entity is an attacker might just mean that one or more insiders (i.e., malicious employees) break the rules, the collusion of two independent entities would require the malicious coordination between the entities, and the consequent continuous communication. Probably, some top managers (maybe CEOs or CISOs) of the two organizations should be involved. When the scope of the malicious activity goes outside the perimeter of a single organization and crosses two independent organizations it is much more difficult to keep the activity secret for a long time. Furthermore, due to the high-level profile of the organizations we are considering in our solution, the risk (in terms of impact) of the disclosure (also *ex-post*) of malicious activities, would likely lead the involved organizations to default, and very serious legal consequences. Assumption **A1** is not applicable to DP-3T because the two TTPs (HF and server) are not independent. Then, in our adversary model, for both protocols, the collusion between HF and the server is allowed.

Assumption **A2** means that, in ZE2-P3T, only a limited number of malicious SIM-cards can be activated provided that TSP is not an attacker. The assumption is quite realistic. At least in the countries in which crime is contrasted, to enable a SIM-card, you should register a real-life person with an ID document. Without the collaboration of the TSP, for an adversary, it would be not simple and very dangerous from a legal point of view to obtain many fake SIM-cards. Even though a black market with stolen/fake SIM-cards could exist, the TSP immediately would detect their massive utilization, thus blocking them.

For both protocols, a huge number of sybil agents can be placed without requiring any registration. For example, by using Bluetooth antennas spread over the territory.

The possible adversaries and their capabilities are:

- The user: they are the only party that operates and captures information available on the territory.
- The server S: it operates only on the basis of the information received from users and HF.

- The health facility HF: it operates only on the basis of the information provided by the user performing a test.

Note that TSP is not included among possible adversaries because, according to Assumption **A1**, TSP could collude only with users. However, even in this case, it can access only to tuples encrypted by the users with the public key of the server. Therefore, it does not have any information associable with contacts.

Now, we formalize all relevant compromises of protocols invalidating privacy and integrity requirements. These implicitly define the security properties of our security model. It is worth noting that we do not consider two kinds of compromises. The first type refers to denial of service compromises. The second type refers to *enumeration attacks* [14]. Enumeration attacks allow a malicious user to estimate the number of positive users. We do not consider this kind of compromise, because it does not appear relevant in terms of impact (indeed, in the literature, no emphasis is given to this attack).

Preliminary, we need to define the *threat-model attributes*, which are: `Attacker_Setting`, `Target`, `Range`.

For each attribute, we also define an order among its values, capturing effort or effect degrees of the compromise.

The domain of `Attacker_Setting` is $\mathcal{A} = \{OU, SU, HU, FC\}$, where

- OU denotes the case in which the attacker is a user not colluding with any other party.
- SU denotes the case in which the attacker colludes with S.
- HU denotes the case in which the attacker colludes with HF.
- FC (which stands for full collusion) denotes the case in which the attacker colludes with S and HF.

We can introduce an order relation into $\mathcal{A}$ as follows (defined in terms of transitive reduction):

$FC \preceq_A SU, FC \preceq_A HU\ , SU \preceq_A OU, HU \preceq_A OU.$

Observe that $\preceq_A$ is partial.

Intuitively, $\preceq_A$ captures the fact that the higher the number of colluding entities, the higher the required effort is.

The domain of `Target` is $\mathcal{T} = \{GV, PV\}$, where

- GV denotes the case in which the victim is a generic user.
- PV denotes the case in which the victim is a user tested positive for the infection who reported their information to the server.

We can introduce an order relation into $\mathcal{T}$ as follows: $PV \preceq_T GV$.

Obviously, if the target is a generic user, the effect of a compromise is greater than the case in which the target is a positive user.

The domain of Range is $\mathcal{R} = \{SC, IC\}$, where

- SC denotes the case in which the compromise can realistically scale to many victims (i.e., the attacker can increase the number of victims at will with no strong barriers).
- IC denotes the case in which the compromise is feasible only if directed to a few number of victims.

We can introduce an order relation into $\mathcal{R}$ as follows: $IC \preceq_R SC$.

Similarly to Target, the effect of a compromise is greater if it involves many victims.

Figure 6.11 summarizes the order relations defined above.

Remark 1. We observe that the three order relations above defined basically refer to set-inclusion/*is-A* hierarchies. In particular, if the attacker performs as a full collusion (FC), it can operate also as server (SU) or Health Facility (HF). Moreover, if it performs as SU or HF, it can operate as a standard user (OU). On the other hand, a positive user (PV) is a generic user (GV), and the set of victims involved in a targeted attack (IC) is a subset of a massive attack (SC).

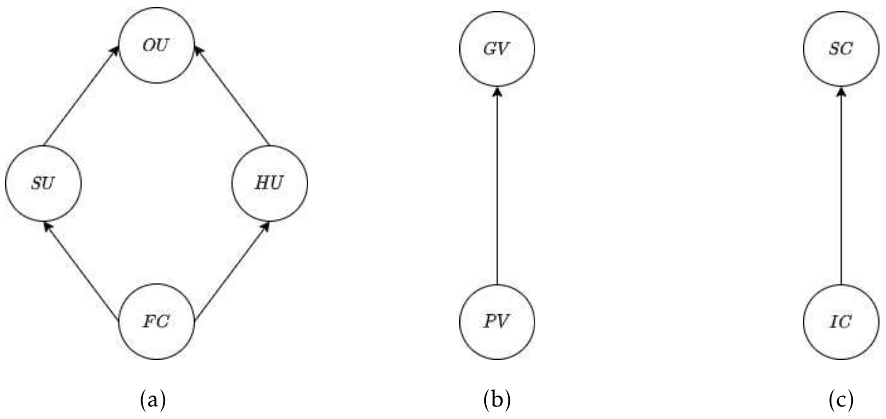


Fig. 6.11: Order relations on $\mathcal{A}$ (a), $\mathcal{T}$ (b), and $\mathcal{R}$ (c)

We focus our security analysis on the following three *Compromises*. Each compromise is defined over $(\mathcal{A}, \mathcal{T}, \mathcal{R})$.

From now on, since the ephemeral identifiers generated by the users in ZE2-P3T are equivalent to the ephemeral identifiers of DP-3T, we use the term *ephemeral ID* to refer to both these pseudonym identities.

Compromise 1: Linkage of ephemeral identifiers

This compromise occurs when the attacker finds out that a given number of ephemeral IDs belong to the same user (not requiring that the real identities are known).

Compromise 2: Contact forgery

This compromise occurs when the attacker is able to forge a contact that never happened between two users with the scope of generating a false alarm.

Remark 2. We remark that, as ephemeral IDs are necessarily pseudonyms unlinkable with real identities, impersonation with a colluding attacker is intrinsically possible both in DP-3T and in ZE2-P3T. Therefore, we do not consider a compromise of type 2 any fake contact forged by an attacker by communicating ephemeral IDs of the colluding attacker in the user-side activity or in the infection-reporting phase.

Compromise 3: Linkage between an ephemeral ID and extra information

This compromise occurs when the attacker is able to link an ephemeral ID of a user with any extra information not provided by the user.

Compromises may happen in different configurations of the attributes of the threat model. A configuration is called *setting*.

Definition 6.1. *A Setting S is a 3-tuple $s = (a, t, r)$, where $a \in \mathcal{A}, t \in \mathcal{T}$, and $r \in \mathcal{R}$. We denote by $\mathcal{S} = \mathcal{A} \times \mathcal{T} \times \mathcal{R}$ the set of all possible settings.*

We have $4 \cdot 2 \cdot 2 = 16$ possible settings.

The partial order relations defined for attributes induces a partial order relation among settings.

Definition 6.2. *Given two settings $S_1 = (a_1, t_1, r_1)$ and $S_2 = (a_2, t_2, r_2)$, we say that $S_1 \preceq_S S_2$ if $a_1 \preceq_A a_2 \land t_1 \preceq_T t_2 \land r_1 \preceq_R r_2$.*

The order relation on $\mathcal{S}$ is represented in Figure 6.12.

In the next definition, we give the notion of *vulnerability of a protocol* to a given compromise in a given setting.

Definition 6.3. *Given a setting S and a compromise C, we say that a protocol P is vulnerable to C in S if there exists an attack in the setting S that leads to the compromise C.*

The following lemma shows that the vulnerability of a protocol to a compromise is monotone with respect to the partial order $\preceq_S$. This lemma gives value to the partial order itself, because it allows us to transitively derive the vulnerability occurrence for each considered protocol and, then, compare them. The partial order can be viewed therefore as a scale of comparison.

Lemma 6.4. *Given two settings S_1 and S_2 and a compromise C such that $S_1 \preceq_S S_2$, if a protocol P is vulnerable to C in S_2, then P is vulnerable to C in S_1.*

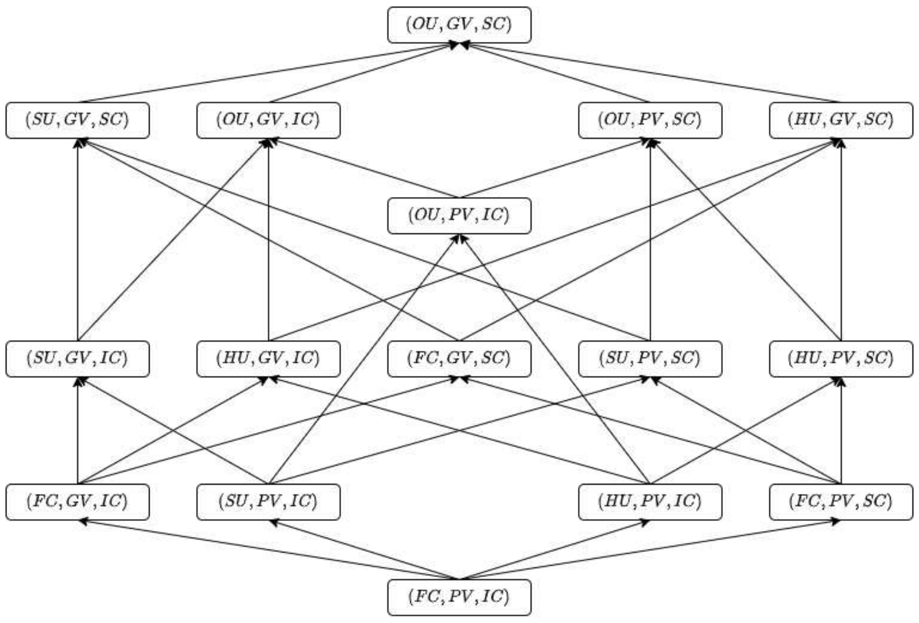


Fig. 6.12: Order relation on $\mathcal{S}$

Proof. The proof immediately follows by **Remark 1** and the fact that $\preceq_S$ is defined on the basis of the three order relations $\preceq_A$, $\preceq_T$, and $\preceq_R$.

As anticipated, the consequence of the above lemma is that, once we have proven the vulnerability of a protocol to a compromise C in a given setting, we have implicitly proven its *derived* vulnerability to C in a number of other settings. Clearly, to capture the whole set of settings in which a protocol is vulnerable to a compromise, we have to refer to settings that are maximal with respect to $\preceq_S$. In other words, if a protocol is vulnerable to a compromise in a maximal setting, then it is vulnerable to all the settings that are lower than this maximal, i.e., the branch of the partial order with this maximal as a top.

This notion of maximality is formalized by the following three definitions.

Definition 6.5. *Given a protocol P, a setting S, and a compromise C, we say that S is a vulnerability-maximal of P (on C) if: (i) P is vulnerable to C in S and (ii) there not exists a setting $S_1 \neq S$ in which P is vulnerable to C such that $S \preceq_S S_1$.*

Definition 6.6. *Given a protocol P and a compromise C, we denote by $\mathcal{V}_{\mathcal{P}}^{\mathcal{C}}$ the set of all the vulnerability-maximals of P on C. $\mathcal{V}_{\mathcal{P}}^{\mathcal{C}}$ is called the skyline set of P (on C).*

Definition 6.7. *Given a protocol P and a compromise C, we denote by $\mathcal{I}_{\mathcal{P}}^{\mathcal{C}}$ the set $\{S \in \mathcal{S} : \exists\, S' \in \mathcal{V}_P^{C}$ s.t $S \preceq_S S'\}$.*

$\mathcal{I}_{\mathcal{P}}^{\mathcal{C}}$ is called the induced set by $\mathcal{V}_{\mathcal{P}}^{\mathcal{C}}$.

In words, the induced set by $\mathcal{V}_{\mathcal{P}}^{\mathcal{C}}$ is the set of settings less or equal to any maximal occurring in $\mathcal{V}_{\mathcal{P}}^{\mathcal{C}}$.

The next proposition states that all the vulnerabilities of a protocol P to a compromise C are induced by the skyline set of P on C. In other words, while so far we referred to the correctness of the vulnerability implication over the partial order, now we state that this mechanism is complete. Therefore, the vulnerabilities of a protocol are all and only those induced by the maximals, i.e., the skyline set.

Proposition 6.8. *Given a protocol P and a compromise C, P is vulnerable to C in a setting S if and only if $S \in \mathcal{I}_{\mathcal{P}}^{\mathcal{C}}$.*

Proof. (**if-part**) Given a set $S \in \mathcal{I}_{\mathcal{P}}^{\mathcal{C}}$, by definition $S \preceq_S S'$ for some $S' \in \mathcal{V}_{\mathcal{P}}^{\mathcal{C}}$. Since P is vulnerable to C in S', due to Lemma 6.4, P is vulnerable to C in S too.

(**only-if-part**) By contradiction suppose that there exists a setting S_x such that P is vulnerable to C in S_x and $S_x \notin \mathcal{I}_{\mathcal{P}}^{\mathcal{C}}$.

Obviously S_x is such that neither $S_x \preceq_S S_y$ nor $S_y \preceq_S S_x$, for any $S_y \in \mathcal{V}_{\mathcal{P}}^{\mathcal{C}}$. Otherwise either S_y would be not a vulnerability-maximal (if $S_y \preceq_S S_x$) or $S_x \in \mathcal{I}_{\mathcal{P}}^{\mathcal{C}}$ (if $S_x \preceq_S S_y$). As $\mathcal{S}$ is finite, this implies that there exists a vulnerability-maximal not included in $\mathcal{V}_{\mathcal{P}}^{\mathcal{C}}$. Therefore, we reached a contradiction.

So far we compared only settings, thanks to the partial order $\preceq_S$. Now, we want to compare protocols in terms of vulnerability degree to a compromise. We say *degree*, by meaning that our comparison should include some *quantitative* feature, to give a measure of how much a protocol is more vulnerable than another protocol. To do this, we first introduce a *weighted* order relation among protocols.

Definition 6.9. *Given two protocols P_1 and P_2, a compromise C, and $k \geq 0$ we say that $P_1 \preceq_k P_2$ (on C), if for each setting $S_x \in \mathcal{V}_{P_2}^{C}$, there exists a setting $S_y \in \mathcal{V}_{P_1}^{C}$ such that $S_x \preceq_S S_y$ and $|\mathcal{I}_{P_1}^{C}| - |\mathcal{I}_{P_2}^{C}| \geq k$.*

With the next corollary, we highlight that even though the above definition is based on skyline sets, actually it regards the whole set of settings in which protocols are vulnerable. Again, this is strictly related to the meaning of maximality, allowing us to express properties at the level of the skyline set, instead of the whole set of settings.

Corollary 6.10. *Given two protocols P_1 and P_2, a compromise C, and $k \geq 0$, it holds that $P_1 \preceq_k P_2$ on C if and only if for each setting S in which P_2 is vulnerable to C, P_1 is vulnerable to C in S too, and there exist at least k distinct settings in which P_1 is vulnerable to C and P_2 not.*

Proof. (**if-part**) By Proposition 6.8, $\mathcal{I}_{P_1}^{C}$ contains all the settings in which P_1 is vulnerable to C. By hypothesis and Proposition 6.8, P_1 is vulnerable to C in each setting $S \in \mathcal{I}_{P_2}^{C}$. Therefore, $\mathcal{I}_{P_2}^{C} \subseteq \mathcal{I}_{P_1}^{C}$. As a consequence, $\mathcal{V}_{P_2}^{C} \subseteq \mathcal{I}_{P_1}^{C}$ as $\mathcal{V}_{P_2}^{C} \subseteq \mathcal{I}_{P_2}^{C}$.

Therefore, by the definition of $\mathcal{I}^C_{P_1}$, for each setting $S_x \in \mathcal{V}^C_{P_2}$ there exists a setting $S_y \in \mathcal{V}^C_{P_1}$ such that $S_x \preceq_S S_y$.

Furthermore, since $\mathcal{I}^C_{P_2} \subseteq \mathcal{I}^C_{P_1}$ and Proposition 6.8, $|\mathcal{I}^C_{P_1}| - |\mathcal{I}^C_{P_2}|$ is the number of settings in which P_2 is vulnerable to C and P_2 not. Therefore, $|\mathcal{I}^C_{P_1}| - |\mathcal{I}^C_{P_2}| \geq k$.

We conclude that $P_1 \preceq_k P_2$.

(**only-if-part**) By hypothesis, for each setting $S_x \in \mathcal{V}^C_{P_2}$, there exists a setting $S_y \in \mathcal{V}^C_{P_1}$ such that $S_x \preceq_S S_y$.

By the definition of $\mathcal{I}^C_{P_2}$, for each setting $S_x \in \mathcal{I}^C_{P_2}$ there exists a setting $S_y \in \mathcal{V}^C_{P_2}$ such that $S_x \preceq_S S_y$.

Then, for each setting $S_x \in \mathcal{I}^C_{P_2}$, there exists a setting $S_y \in \mathcal{V}^C_{P_1}$ such that $S_x \preceq_S S_y$ and then $\mathcal{I}^C_{P_2} \subseteq \mathcal{I}^C_{P_1}$.

Therefore, P_1 is vulnerable to C in each S in which P_2 is vulnerable to C. Finally, since $|\mathcal{I}^C_{P_1}| - |\mathcal{I}^C_{P_2}| \geq k$, there are at least k settings in which P_1 is vulnerable to C and P_2 not.

We are now ready to compare the security of our protocol with the two designs of DP-3T, with respect to the three compromises. We remark that this comparison implicitly includes also GAEN, because GAEN, from the protocol point of view, consists of DP-3T with Low-Cost design. This is obtained by means of the following three theorems. Therein, we denote by LD the Low-Cost design of DP-3T, by UD the Unlinkable design of DP-3T, and by ZP our protocol. We compare the protocols per compromise.

We start by considering the compromise C_1. The next theorem states that, on compromise C_1, Low-Cost DP-3T is more vulnerable (with *degree* 6) than our protocol, and Unlinkable DP-3T is more vulnerable (with *degree* 4) than our protocol.

Theorem 6.11. *LD* $\preceq_6$ *ZP* $\wedge$ *UD* $\preceq_4$ *ZP on the compromise* C_1.

Proof. First, we prove $\mathcal{V}^{C_1}_{ZP} = \{(SU, GV, IC), (SU, PV, SC)\}$.

To do this, first (i) we prove that ZP is vulnerable to C_1 in (SU, GV, IC) and (SU, PV, SC).

Then (ii), we prove that ZP is not vulnerable to C_1 in any setting greater than (SU, GV, IC) or (SU, PV, SC).

Finally (iii), we prove that ZP is not vulnerable to C_1 in any other incomparable setting w.r.t (SU, GV, IC) and (SU, PV, SC).

We proceed by proving (i). Consider the setting (SU, GV, IC). Regarding the compromise C_1, this setting means that the attacker colludes with the server to link the ephemeral IDs of a limited number of generic users.

Observe that the information received by the server from a user cannot allow the guessing of the centroid sent (in hashed form) by the user, provided that the salt sent by the TSP remains unknown to the server. Otherwise, a successful dictionary-based attack is possible. On this basis, the server can link different tuples (and then the included ephemeral IDs) corresponding to the same user also by using attacks based on trajectories. However, in the considered setting, this may happen by selecting a victim and detecting the salt sent from the TSP in the zone in which the victim is located. Therefore, ZP is vulnerable to C_1 in (SU,GV,IC).

Regarding (SU,PV,SC) (i.e., the attacker colludes with the server to link the ephemeral IDs of a scalable number of positive users), it is easy to see that ZP is vulnerable to C_1 in (SU,PV,SC). In fact, each user tested positive for the disease sends all their ephemeral IDs at once to the server. Therefore, the proof of (i) is concluded.

Now, consider (ii). The settings greater than (SU,GV,IC) or then (SU,PV,SC) are {(SU,GV,SC), (OU,GV,IC), (OU,GV,SC),(OU,PV,SC)}.

The setting (SU,GV,SC) differs from (SU,GV,IC) only in the fact that the former involves a scalable number of victims. According to the reasoning done for the setting (SU,GV,IC), (SU,GV,SC) would be vulnerable only if a scalable number of salts given by the TSP are known to the server.

Indeed, without TSP salts, the tuples owned by the server cannot be associated with any information not sent by the user. But, receiving a scalable number of salts from different places in the territory is impossible according to Assumption **A2**. Thus, ZP is not vulnerable to C_1 in (SU,GV,SC).

In both the settings (OU,GV,IC) and (OU,PV,SC), the attacker does not collude with the server. The only way for the attacker to obtain the ephemeral IDs of other users is at the end of the contact-detection phase, when the attacker receives a list of ephemeral IDs belonging to users who entered into contact with a positive user. However, the attacker can not distinguish if some of them belong to the same user or to different users. Therefore, ZP is not vulnerable to C_1 in (OU,GV,IC) and (OU,PV,SC).

Due to Lemma 6.4, since ZP is not vulnerable to C_1 in (SU,GV,SC), then it is not vulnerable to C_1 in (OU,GV,SC). Therefore, the proof of (ii) is concluded.

Finally, consider (iii). The settings incomparable with respect to (SU,GV,IC) and (SU,PV,SC) are {(OU,PV,IC), (HU,GV,SC), (HU,GV,IC), (FC,GV,SC), (HU,PV,SC), (HU,PV,IC)}.

Consider the setting (FC,GV,SC) i.e., the attacker colludes with the server and HF to link the ephemeral IDs of a scalable number of users. HF does not receive any ephemeral ID from users. Then, the collusion with HF does not provide any advantage to the attacker with respect to the setting (SU,GV,SC). Therefore, ZP is not vulnerable to C_1 in (FC,GV,SC).

Regarding the setting (OU,PV,IC), since the server does not collude, the attacker can obtain the ephemeral IDs only during the contact-detection phase but it is unable to link them. Therefore, ZP is not vulnerable to C_1 in (OU,PV,IC).

Finally, since the collusion with HF does not provide any advantage to the attacker, the settings (HU,GV,SC), (HU,GV,IC), (HU,PV,SC), (HU,PV,IC) are equivalent to the settings (OU,GV,SC), (OU,GV,IC), (OU,PV,SC), (OU,PV,IC), respectively, in which ZP is not vulnerable. This concludes the proof of (iii).

At this point, since $\mathcal{V}_{ZP}^{C_1}=\{(SU,GV,IC),(SU,PV,SC)\}$, the induced set by $\mathcal{V}_{ZP}^{C_1}$ is $\mathcal{I}_{ZP}^{C_1}=\{(SU,GV,IC),(SU,PV,SC),(FC,GV,IC),(SU,PV,IC),(FC,PV,SC),(FC,PV,IC)\}$ and $|\mathcal{I}_{ZP}^{C_1}|=6$.

Now, consider the protocol LD. To prove that LD $\preceq_6$ ZP, we have to show that (i) there exist $S_1,S_2\in\mathcal{V}_{LD}^{C_1}$ such that $(SU,GV,IC)\preceq_S S_1$ and $(SU,PV,SC)\preceq_S S_2$, and (ii) $|\mathcal{I}_{LD}^{C}|-|\mathcal{I}_{ZP}^{C}|\geq 6$.

We prove (i).

LD is vulnerable to C_1 in (OU,GV,IC). In fact, the attacker, without colluding with the server, can link the ephemeral IDs released in broadcast (in plaintext) by a number of target users.

Due to Proposition 6.8, $(OU,GV,IC)\in\mathcal{I}_{LD}^{C_1}$, then there exist $S_1\in\mathcal{V}_{LD}^{C_1}$ such that $(SU,GV,IC)\preceq_S(OU,GV,IC)\preceq_S S_1$.

LD is also vulnerable to C_1 in (OU,PV,SC). In fact, each user receives the secret key of each positive user and, locally, computes the ephemeral IDs of the latter.

Due to Proposition 6.8, $(OU,PV,SC)\in\mathcal{I}_{LD}^{C_1}$, then there exist $S_2\in\mathcal{V}_{LD}^{C_1}$ such that $(SU,PV,SC)\preceq_S(OU,PV,SC)\preceq_S S_2$. Then, the proof of (i) is given.

Now, consider (ii).

The set $\mathcal{J}_{LD}^{C_1}=\{(OU,GV,IC),(OU,PV,SC),(OU,PV,IC),(SU,GV,IC),(HU,GV,IC),(SU,PV,SC),(HU,PV,SC),(FC,GV,IC),(SU,PV,IC),(HU,PV,IC),(FC,PV,SC),(FC,PV,IC)\}\subseteq\mathcal{I}_{LD}^{C_1}$, since each element $S\in\mathcal{J}_{LD}^{C_1}$ is such that either $S\preceq_S(OU,GV,IC)$ or $S\preceq_S(OU,PV,SC)$ and $(OU,GV,IC),(OU,PV,SC)\in\mathcal{I}_{LD}^{C_1}$.

Therefore, $|\mathcal{I}_{LD}^{C_1}|\geq|\mathcal{J}_{LD}^{C_1}|=12$ and then $|\mathcal{I}_{LD}^{C_1}|-|\mathcal{I}_{ZP}^{C_1}|\geq 6$. The proof of (ii) is given.

Finally, we consider the protocol UD. The reasoning is the same as LD.

To prove that UD $\preceq_4$ ZP, we have to show that (i) there exist $S_1,S_2\in\mathcal{V}_{UD}^{C_1}$ such that $(SU,GV,IC)\preceq_S S_1$ and $(SU,PV,SC)\preceq_S S_2$, and (ii) $|\mathcal{I}_{UD}^{C}|-|\mathcal{I}_{ZP}^{C}|\geq 4$.

We prove (i).

Similarly to LD, UD is vulnerable to C_1 in (OU,GV,IC). In fact, the attacker, without colluding with the server, can link the ephemeral IDs released in broadcast (in plaintext) by a number of target users.

Due to Proposition 6.8, $(OU,GV,IC)\in\mathcal{I}_{UD}^{C_1}$, then there exist $S_1\in\mathcal{V}_{UD}^{C_1}$ such that $(SU,GV,IC)\preceq_S(OU,GV,IC)\preceq_S S_1$.

Similarly to ZP, UD is vulnerable to C_1 in (SU,PV,SC). In fact, in UD, each positive user sends a list of seeds from which the server retrieves the ephemeral IDs and then can link them.

Due to Proposition 6.8, $(SU,PV,SC) \in \mathcal{I}_{UD}^{C_1}$, then there exist $S_2 \in \mathcal{V}_{UD}^{C_1}$ such that $(SU,PV,SC) \preceq_S S_2$. Therefore, the proof of (i) is concluded.

Consider (ii).

The set $\mathcal{J}_{UD}^{C_1} = \{(OU,GV,IC),(OU,PV,IC),(SU,GV,IC),(HU,GV,IC),(SU,PV,SC),(FC,GV,IC),(SU,PV,IC),(HU,PV,IC),(FC,PV,SC),(FC,PV,IC)\} \subseteq \mathcal{I}_{UD}^{C_1}$ since each element $S \in \mathcal{J}_{UD}^{C_1}$ is such that either $S \preceq_S (OU,GV,IC)$ or $S \preceq (SU,PV,SC)$ and $(OU,GV,IC),(SU,PV,SC) \in \mathcal{I}_{UD}^{C_1}$.

Therefore, $|\mathcal{I}_{UD}^{C_1}| \geq \mathcal{J}_{UD}^{C_1} = 10$ and then $|\mathcal{I}_{UD}^{C_1}| - |\mathcal{I}_{ZP}^{C_1}| \geq 4$. This concludes the proof of (ii) and then the proof of the Theorem too.

Now, we compare the security of ZE2-P3T with DP-3T with respect to the compromise C_2. The next theorem state that, on compromise C_1, Low-Cost DP-3T is more vulnerable (with *degree* 4) than our protocol, and Unlinkable DP-3T is more vulnerable (with *degree* 4) than our protocol.

Theorem 6.12. *$LD \preceq_4 ZP \wedge UD \preceq_4 ZP$ on the compromise C_2.*

Proof. First, we prove $\mathcal{V}_{ZP}^{C_2} = \{(SU,GV,SC),(OU,GV,IC)\}$.

To do this, first (i) we prove that ZP is vulnerable to C_2 in (SU,GV,SC) and (OU,GV,IC).

Then (ii), we prove that ZP is not vulnerable to C_2 in any setting greater than (SU,GV,SC) or (OU,GV,IC).

Finally (iii), we prove that ZP is not vulnerable to C_2 in any other setting incomparable w.r.t (SU,GV,SC) and (OU,GV,IC).

We proceed by proving (i). Regarding the compromise C_2, the setting (SU,GV,SC) means that the attacker colludes with the server to forge fake contacts among a scalable number of generic users. In ZP, the server can forge fake contact tuples during the contact-detection phase by using the ephemeral IDs received by the users during the user-side activity. Then, ZP is vulnerable to C_2 in all the settings where the first component is SU or FC. In particular, ZP is vulnerable to C_2 in (SU,GV,SC).

Consider now the setting (OU,GV,IC). It means that the attacker tries to forge (without colluding with the server) fake contacts among a restricted number of generic users. Considering a single victim, the attacker can physically follow the victim (without entering into contact), capture the salt from the TSP, and compute the same digest of the victim (by using the same centroid). At this point, the attacker is able to build a restricted tuple that matches (same time slot and same digest) with that provided by the victim. Then, ZP is vulnerable to C_2 in (OU,GV,IC).

The proof of (i) is given.

Now, consider (ii). The only setting greater than (SU,GV,SC) or (OU,GV,IC) is (OU,GV,SC) (that does not include collusion with the server). To forge a fake contact, as the server is trusted, the attacker should be able to send the server fake information that will determine a (forged) contact. Recall that a contact between two ephemeral IDs is detected on the basis of the timestamp and digest. Therefore, the attacker would tamper one of the above pieces of information. First, consider the ephemeral IDs. As they are not exchanged among users, the only possibility is that two colluding attackers agree by switching their ephemeral IDs with the aim to jeopardize either the user-side activity or infection-reporting phase by reporting the ephemeral of the other user. However, due to **Remark** 2, we are not in the case of the compromise C_2.

Consider now the remaining information (i.e., timestamp and digest). Since the server checks the current timestamp is not too greater than the timestamp provided by the users, it is not possible to tamper such information. Regarding the digests, as observed in Theorem 6.11, they cannot be tampered without knowing the salts provided by the TSP. Then, by Assumption **A2**, and by considering that when an infection is reported the server takes into consideration only one ephemeral per time slot, it is not possible to forge digests on a large scale even in case of high concentration of people (for which a few salts would be enough for the attacker). Therefore, ZP is not vulnerable to C_2 in (OU,GV,SC). This concludes the proof of (ii).

Finally, consider (iii). The settings incomparable with respect to (SU,GV,SC) and (OU,GV,IC) are $\{(OU,PV,SC),(HU,GV,SC),(HU,PV,SC)\}$. Since the server does not collude, by applying a reasoning similar to that used in the proof of (ii), by Assumption **A2**, it is not possible to forge contact on a large scale. Then ZP is not vulnerable to C_2 in (OU,PV,SC), (HU,GV,SC), and (HU,PV,SC). Therefore, the proof of (iii) is concluded.

At this point, since $\mathcal{V}_{ZP}^{C_2} = \{(SU,GV,SC),\ (OU,GV,IC)\}$, the induced set by $\mathcal{V}_{ZP}^{C_2}$ is $\mathcal{I}_{ZP}^{C_2} = \{(SU,GV,SC), (OU,GV,IC), (OU,PV,IC), (SU,GV,IC), (HU,GV,IC), (FC,GV,SC), (SU,PV,SC), (FC,GV,IC), (SU,PV,IC), (HU,PV,IC), (FC,PV,SC), (FC,PV,IC)\}$ and $|\mathcal{I}_{ZP}^{C_2}| = 12$.

Regarding both LD and UD, consider the setting (OU,GV,SC). The attacker can detect the ephemeral IDs released in broadcast by the users (potentially, a huge number) and send them other ephemeral IDs detected by other users through BLE antennas spread over the territory.

Then, both LD and UD are vulnerable to C_2 in (OU,GV,SC).

Each setting $S \in \mathcal{S}$ is such that $S \preceq_S (OU,GV,SC)$, therefore we have that: $\mathcal{V}_{LD}^{C_2} = \mathcal{V}_{UD}^{C_2} = \{(OU,GV,SC)\}$ and $|\mathcal{I}_{LD}^{C_2}| = |\mathcal{I}_{UD}^{C_2}| = |\mathcal{S}| = 16$. Therefore, LD $\preceq_4$ ZP $\wedge$ UD $\preceq_4$ ZP.

The proof of the theorem is concluded.

In the next theorem, we state the superiority of ZE2-P3T on the security with respect to the compromise C_3. Indeed, on compromise C_1, Low-Cost DP-3T is more vulnerable (with *degree* 12) than our protocol, and Unlinkable DP-3T is more vulnerable (with *degree* 12) than our protocol.

Theorem 6.13. *LD $\preceq_{12}$ ZP $\wedge$ UD $\preceq_{12}$ ZP on the compromise C_3.*

Proof. First, we prove $\mathcal{V}_{ZP}^{C_3} = \{(SU,GV,IC)\}$.

To do this, first (i) we prove that ZP is vulnerable to C_3 in (SU,GV,IC).

Then (ii), we prove that ZP is not vulnerable to C_3 in any setting greater than (SU,GV,IC).

Finally (iii), we prove that ZP is not vulnerable to C_3 in any other setting incomparable w.r.t (SU,GV,IC).

We proceed by proving (i). In the setting (SU,GV,IC), the server colludes with the attacker in order to associate extra information (e.g., the position) with the ephemeral IDs of a limited number of generic users.

The reasoning is the same as that done in Theorem 6.11 for the setting (SU,GV,IC). The attacker may select a victim and detect the salts sent from the TSP in the zones in which the victim is located. This way, the server can compute the digests, identify the (restricted) tuples sent by the user containing the ephemeral IDs, and associate extra information to these latter. Thus, ZP is vulnerable to C_3 in (SU,GV,IC). The proof of (i) is given.

Now, consider (ii). The settings greater than (SU,GV,IC) are $\{(SU,GV,SC)$, (OU,GV,IC), $(OU,GV,SC)\}$. The setting (SU,GV,SC) differs from (SU,GV,IC) only because the former involves a scalable number of victims. According to the reasoning done for the setting (SU,GV,IC), (SU,GV,SC) would be vulnerable only if a scalable number of salts given by the TSP are known to the server. Anyway, receiving a scalable number of salts from different places in the territory is impossible according to Assumption **A2**. Thus, ZP is not vulnerable to C_3 in (SU,GV,SC).

In the setting (OU,GV,IC), the server does not collude. As already discussed in Theorem 6.11, an attacker non-colluding with the server can obtain the ephemeral IDs of other users only at the end of the contact-detection phase but no extra information is associated with them. Then, there is no way for an attacker to link these ephemeral IDs with other external information. Thus, ZP is not vulnerable to C_3 in (OU,GV,IC).

Due to the Lemma 6.4, since ZP is not vulnerable to C_3 in (SU,GV,SC), then it is not vulnerable to C_3 in (OU,GV,SC). Therefore, the proof of (ii) is concluded.

Finally, consider (iii). The settings incomparable with respect to (SU,GV,IC) are $\{(OU,PV,SC)$, (HU,GV,SC), (OU,PV,IC), (HU,GV,IC), (FC,GV,SC), (SU,PV,SC), (HU,PV,SC), (HU,PV,IC), $(FC,PV,SC)\}$.

Consider the setting (FC,PV,SC), where HF and the server collude with the attacker in order to associate extra information to an ephemeral ID of a scalable number of positive users. The ephemeral IDs are released by the users or (i) during the user-side activity (previously they perform the test in the HF) or (ii) during the infection-reporting phase. Regarding (i), HF is not involved and the positive users perform as generic users, then the setting (FC,PV,SC) is equivalent to (SU,GV,SC) and due to Assumptions **A2**, ZP is not vulnerable to C_3 in (SU,GV,SC)). In case (ii), the users upload their tuples containing the ephemeral IDs to the server at a non-predictable instant, and then, no information can be associated directly with them. However, some information can be associated indirectly through the collusion with the HF. Even though HF knows extra information about a positive user (e.g., the identity) and communicates it to the server, there is no way to link such information with the ephemeral IDs of the user. In fact, the blind signature scheme ensures that the credential $\sigma(M)$, used as authorization code to upload the ephemeral IDs, is unlinkable for HF with the message provided by the user to obtain the above credential. Thus, there is no way for the attacker to link the uploaded ephemeral IDs to any associated extra information. Then, ZP is not vulnerable to C_3 in (FC,PV,SC).

Due to the Lemma 6.4, since ZP is not vulnerable to C_3 in (FC,PV,SC), then it is not vulnerable to C_3 in (FC,GV,SC), (SU,PV,SC), (HU,PV,SC), (HU,GV,SC), and (OU,PV,SC).

Consider the setting (HU,PV,IC). Again, an attacker non-colluding with the server can obtain the ephemeral IDs only at the end of the contact-detection phase but no extra information is associated with them. There is no way for an attacker to link these ephemeral IDs with other external information. Thus, ZP is not vulnerable to C_3 in (HU,PV,IC).

Due to the Lemma 6.4, since ZP is not vulnerable to C_3 in (HU,PV,IC), then it is not vulnerable to C_3 in (HU,GV,IC), and (OU,PV,IC). The proof of (iii) is given.

At this point, since $\mathcal{V}_{ZP}^{C_3} = \{(SU,GV,IC)\}$, the induced set by $\mathcal{V}_{ZP}^{C_3}$ is $\mathcal{I}_{ZP}^{C_3} = \{(SU,GV,IC), (FC,GV,IC), (SU,PV,IC), (FC,PV,IC)\}$ and $|\mathcal{I}_{ZP}^{C_1}| = 4$.

Regarding both LD and UD, consider the setting (OU,GV,SC). It is easy to see that both LD and UD are vulnerable to C_3 in (OU,GV,SC). In fact, the attacker, without colluding with the server, can associate extra information with each ephemeral ID released in broadcast (in plaintext) by the users and this can be done

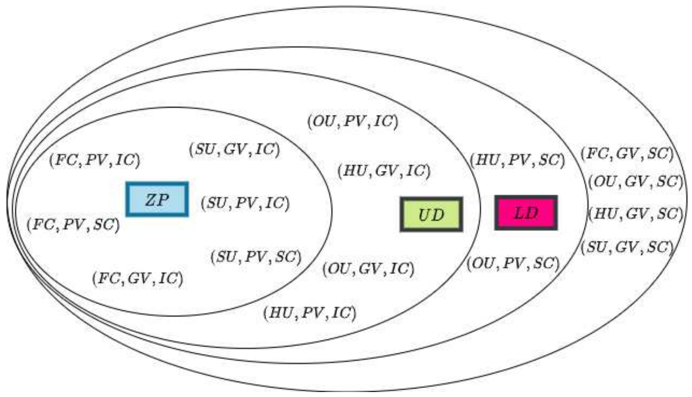


Fig. 6.13: Visual comparison on compromise C_1.

against a scalable number of users, without violating limits imposed by Assumption **A2** (i.e., the intervention of a non-attacker TTP).

Each setting $S \in \mathcal{S}$ is such that $S \preceq_S (OU, GV, SC)$, therefore we have that: $\mathcal{V}_{LD}^{C_3} = \mathcal{V}_{UD}^{C_3} = \{(OU, GV, SC)\}$ and $|\mathcal{I}_{LD}^{C_3}| = |\mathcal{I}_{UD}^{C_3}| = |\mathcal{S}| = 16$. Therefore, LD $\preceq_{12}$ ZP $\wedge$ UD $\preceq_{12}$ ZP.

The proof then is concluded.

In Figures 6.13, 6.14, and 6.15 we provide, coherently with Theorems 6.11,6.12, and 6.13, a visual representation of the improvements in terms of security given by our protocol with respect to DP-3T. Therein, all the settings of S are represented and it appears evident that the sets of settings in which the latter protocol is vulnerable (in both designs Low-cost and Unlinkable) are supersets of the vulnerable settings for our protocol, for any compromise. For example, from Figure 6.14, we see that both LD and UD are vulnerable to C_2 in all the settings of S. Note that, Figure 6.13, corresponding to Theorem 6.11, represents the set of all the settings in which ZP is vulnerable. Instead, for UD and LD, the theorem singles out a subset (possibly improper) of settings in which they are vulnerable. Indeed, we only need to have a *lower bound* difference in terms of security between ZP and DP-3T, not a complete characterization of the security of DP-3T. The same reason cannot be applied to Theorems 6.12, and 6.13, and then to Figures 6.14 and 6.15, because in that case all the settings of S are covered.

In the following part of the section, we show a number of attacks already reported in the literature and contextualize them within the theoretical framework introduced earlier. Note that this description has a practical relevance, as witnesses, through concrete exploits, how the existing techniques perform. This description has not the aim to further demonstrate something about the security of our approach, which is completely addressed so far. We remark that, given a setting, when a protocol is vulnerable to a compromise in this setting, this means that there exists at least

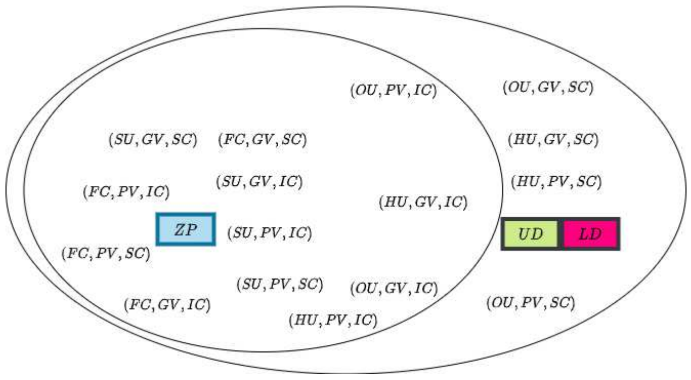


Fig. 6.14: Visual comparison on compromise C_2.

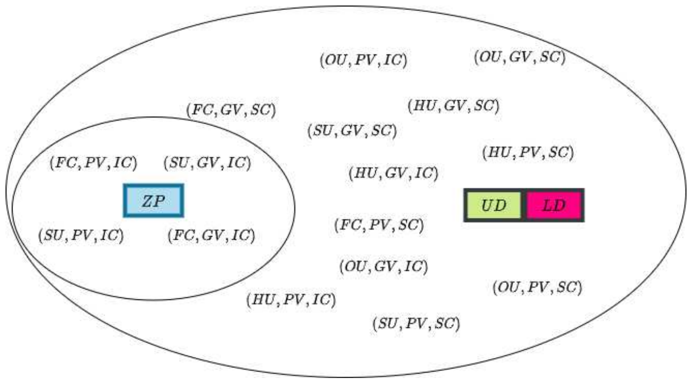


Fig. 6.15: Visual comparison on compromise C_3.

one attack witnessing such a vulnerability. Therefore, it can happen that for other attacks in the same setting, this protocol is not vulnerable.

Paparazzi Attack [23]: The purpose of the attack is to put on a mass tracking system on infected users. The attacker can be anyone and no collusion with the server or HF is required. The attack leads to the compromises C_1 and C_3 and the setting considered is (OU, PV, SC). It is performed as follows. First, the attacker places several passive BLE devices through the territory in order to collect the ephemeral IDs of other users located in the proximity of such devices. Moreover, it records the time and the location where such identifiers are received and, possibly, other information about the users. This attack works only on the Low-Cost Design of DP-3T. In fact, when a user U results infected, they send a single secret seed SK to the server which, in turn, broadcasts it (together with the seeds of other positive users) to all the users, including the attacker.

Starting from SK, the attacker is able to generate all the ephemeral identifiers of U and, for each of them, the attacker retrieves the data (time, location, etc.) stored when U passed in the proximity of the passive devices. The linkage of these data allows the attacker to track the user.

We suppose that, for example, the passive devices associate the coordinates of the users with their ephemeral IDs. By linking the IDs of a positive user, the attacker retrieves a list of pairs of coordinates (each pair is associated with an ephemeral ID) that allow identifying the trajectory of the positive user.

This attack does not work on the Unlinkable design of DP-3T since the infected user U sends the seeds to generate the ephemeral IDs to the server, but this latter does not broadcast such seeds to all users. Instead, the server generates all the ephemeral identifiers of U and adds them to the Cuckoo filter, so that the attacker cannot link them as belonging to the same user.

Regarding ZP, the ephemeral IDs are not exchanged through BLE, but they are sent directly to the server (encrypted with its public key). Therefore, there is no way for the attacker to associate extra information to these IDs and the attack cannot be performed.

We emphasize that the above discussion is perfectly aligned with Theorems 6.11 and 6.13. In fact, the attack works with LD that is vulnerable to C_1 and C_3 in the setting (OU, PV, SC) and it does not work on ZP that is not vulnerable neither to C_1 nor to C_3 in (OU, PV, SC). Regarding UD, it is vulnerable to C_3 in the setting (OU, PV, SC) but we did not prove that it is vulnerable to C_1 in (OU, PV, SC). Observe that Paparazzi can be viewed as an implementation of the most general concept of *linkage attack* [14]. Obviously, the relationship described here between Paparazzi and our security framework can be directly applied to any linkage attack.

Orwell Attack [23]: The attack is very similar to the Paparazzi attack. The difference is that the attacker colludes with the server and, consequently, they can access to all the information therein stored. This makes both Unlinkable and Low-Cost designs of DP-3T vulnerable to the Orwell attack. The attack leads to the compromises C_1 and C_3 and the setting considered is (SU, PV, SC).

Regarding the Low-Cost design, trivially, the Orwell attack without the collusion with the server is equivalent to the Paparazzi attack to which the Low-Cost design is vulnerable.

Regarding the Unlinkable design, what is missing to the attacker to perform the Paparazzi attack is the linkage between the ephemeral identifiers of the same positive user. In the Orwell attack, this linkage can be provided by the server and the attack can be performed.

In particular, the server knows the seeds uploaded by an infected user, and thus, it is able to link the ephemeral identifiers generated by such seeds. Once obtained the ephemeral IDs, the attack performs exactly as the Paparazzi attack, and the user is tracked through the information stored when they passed near the passive devices.

We show that such an attack is not possible in ZP. Indeed, as discussed in the Paparazzi attack, the ephemeral IDs (or other identifying information) are not exchanged between users but are sent directly to the server. However, from the considerations of Theorem 6.13, the server is not able to identify the tuples sent by the users in a massive fashion due to Assumption **A2**. Then, even though the attacker captures information of users in several places, it can not associate them with their ephemeral IDs.

With reference to Theorems 6.11 and 6.13, this attack works on LD and UD and, indeed, they are vulnerable to both C_1 and C_3 in (SU, PV, SC). Instead, ZP is not vulnerable to such an attack as we expected since it is not vulnerable to C_3 in (SU, PV, SC).

Brutus attack [23]: In this attack, the attacker colludes with the health facility HF and the server to find out the mapping between pseudonyms and real identities of infected users during the infection-reporting phase. The attack leads to the compromises C_1 and C_3 and the setting considered is (FC, PV, SC).

It is an exploit of the authorization mechanism with which infected users communicate their status to the server. DP-3T (both the designs) proposes three different authorization mechanisms but they are, essentially, based on an authorization code released by HF to the user and to the server.

Obviously, HF knows the real identity of the user performing the test and, therefore, knows the mapping between the identity and the authorization code.

However, when the infected users upload their ephemeral IDs (through a single seed for the Low-Cost design or through a set of seeds for the Unlinkable design), the server knows the mapping between such IDs and the authorization code. By merging the two mappings, the attacker is able to associate the real identity of the users with their ephemeral IDs.

In ZP, the authorization code is replaced by M which cannot be linked by HF to the message submitted by the user to obtain the signature, thanks to the blind signature mechanism.

Server-side, when the server verifies the signature, it is sure that a generic user is enabled by HF to upload its ephemeral ids, but it does not know who the user is.

Thus, both HF and the server cannot link M to the real identity of the user. In conclusion, ZP is not vulnerable to Brutus attack.

With reference to Theorems 6.11 and 6.13, this attack works on LD and UD and, indeed, they are vulnerable to both C_1 and C_3 in (FC, PV, SC). ZP is not vulnerable to such an attack since, even though it is vulnerable to C_1 in (FC, PV, SC), it is not vulnerable to C_3 in (FC, PV, SC).

Gossip attack [23]: The objective of this attack is to provide any evidence about an encounter with an infected user before discovering their positiveness to the infection. No collusion with the server or HF is required. The attack leads to the compromises C_1 and C_3 and the setting considered is (OU, PV, IC).

The attack works on both the designs of DP-3T due to the exchange of the ephemeral IDs.

The attacker intercepts the ephemeral IDs of the other users (it can use BLE passive devices as in the Paparazzi attack) and stores them in a repository that provides sufficient guarantees on the time t in which data are stored. For example, we can think of blockchain.

If any of such users is tested positive, they upload their ephemeral IDs and the attacker is able to prove an encounter with the infected user at a given instant t before the upload of the IDs.

It can be viewed as a security flaw because it is a misuse of the system for an unintended scope, potentially threatening privacy and exploitable for disputes.

In ZP, this attack is not possible since users do not exchange any ephemeral ID.

Again, these results are compliant with Theorems 6.11 and 6.13. In fact, LD and UD are vulnerable to both C_1 and C_3 in (OU, PV, IC) and ZP is not vulnerable neither to C_1 nor to C_3 in (OU, PV, IC).

Matteotti attack [23]: In this attack, the attacker colludes with the server in order to build a fake contact between a user victim and a positive user. The result aimed by the attacker is to damage the victim by enforcing their quarantine (or other consequent actions). The attack leads to the compromises C_2 the setting considered is (SU, GV, IC). We denote by U_v the user victim and by U_s any user which entered into contact with U_v exchanging its ephemeral IDs. The attack works only on the Unlinkable design of DP-3T.

The attacker captures (e.g., through BLE passive devices) the ephemeral IDs generated by U_s when they encounter U_v and sends them to the server. The latter inserts such identifiers in the Cuckoo filter, even if U_s is not positive, so that, when U_v checks the filter, they are wrongly alerted.

In the Low-Cost design, the ephemeral IDs of a positive user are generated directly by the other users through a single secret seed (provided by the infected user) that the attacker or the server are not able to recover. Therefore, the attack does not work.

In ZP, the server, colluding with a partner located in the same zone as the victim, can retrieve the salt of the TSP, identify the ephemeral ID of the victim, and add it in the list of the positive ephemeral IDs. Thus, ZP is susceptible to this attack.

In summary, this attack works on ZP and UD and they are vulnerable to C_2 in (SU, GV, IC). However, even though such an attack does not work on LD, it is anyway vulnerable to C_2 in (SU, GV, IC) since, as shown in Theorem 6.12, it is vulnerable to C_2 in (OU, GV, SC).

The following two attacks are in the setting (OU, GV, SC) and both LD and UD are vulnerable.

Missile attack [41]: Similarly to the Matteotti attack, in the Missile attack the objective is to alert other users with a fake contact with a positive user. The attack leads to the compromises C_2 the setting considered is (OU, GV, SC). This time, the attacker colludes with a positive user U to obtain their ephemeral IDs before U communicates them to the server. No collusion with the server is required.

The attacker can use a Bluetooth amplifier transmitter to send (like a *missile*) the ephemeral identifiers of U to other users even very far from the positive user and so, not at risk.

Subsequently, when such ephemeral IDs are reported to the server, the other users are wrongly alerted. Since the user U colluding with the attacker is really positive, in the Low-Cost design, U can upload the secret seed that generates their ephemeral IDs, thus this attack works on both the designs of DP-3T.

On the contrary, ZP does not suffer from this attack since no identifier is exchanged through BLE.

These results are compliant with Theorem 6.12 that shows that LD and UD are vulnerable to C_2 in all the settings, and then in (OU, GV, SC), while ZP is not vulnerable to C_2 in (OU, GV, SC).

Fregoli (or Relay) attack: This attack has been reported independently in [168] and [41] (with a slight temporal shift) and is one of the most serious threats of an ephemeral-based contact-tracing protocol.

In this attack, the attacker collects the ephemeral IDs of other users and sends them in broadcast in place of their own. The purpose is to simulate fake contacts between users. The attack leads to the compromises C_2 and the setting considered is (OU, PV, SC). This way, a generic user U maintains a list of ephemeral IDs belonging to users U never met. If any of such users is positive, U is wrongly alerted. This is then an impersonation attack, as its name evokes, being Fregoli one of the major *quick-change* artists of the story. This attack is more effective when a Bluetooth amplifier is used as in the Missile attack. Again, the vulnerability is the exchange of ephemeral IDs between users, thus the attack works on both the design of DP-3T but not on ZP.

These results are compliant with Theorem 6.12 that shows that LD and UD are vulnerable to C_2 in all the settings, and then in (OU, GV, SC), while ZP is not vulnerable to C_2 in (OU, GV, SC).

Battleship attack [41]: This attack applies to the localization-based solutions.

The attacker, colluding with the server, tries to identify the position of a huge number of users to track them. The compromises involved are C_1 and C_3 since if the attacker knows the position of a user at a given instant, it is able to identify the tuples sent by the user and then to link the ephemeral IDs they contain and to associate them with extra information (e.g., the position of the user). The setting considered is (SU, GV, SC).

Since in DP-3T, no information about the position is sent to the server, the attack cannot be performed. On the contrary, any standard GPS-based solution is affected by this problem. Therefore, it is important to check what happens for our protocol. In ZP, the only information about the position of the users is reported in the digests. The only way to reverse them (and discover the centroids in which the users are located) is to know the salts sent by the TSP.

As explained in the Orwell Attack, even though the attack can be performed on a limited number of users, to put on a mass tracking system is infeasible.

In summary, this attack does not work on LD, UD, and ZP.

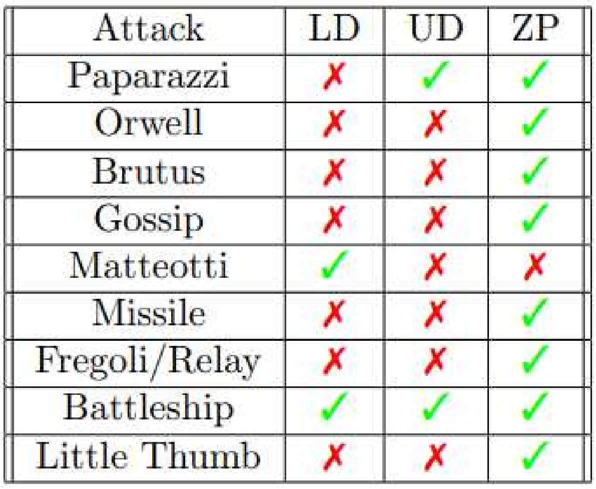
Little Thumb attack [170]: This attack is due to the practical implementation of DP-3T based on GAEN. In fact, in order to avoid the linking of the ephemeral IDs, they have to change simultaneously with the MAC addresses of the BLE messages sent in broadcast. However, if they are not perfectly synchronized to link the ephemeral IDs becomes possible. In fact, suppose that the ephemeral id changes before the MAC address. In this case, the user sends in broadcast two consecutive messages with the same MAC address and two different ephemeral IDs that can be linked. Once the ephemeral IDs are linked, we have all the drawbacks of the Paparazzi and Orwell attacks. The attack leads to the compromises C_1 and the setting considered is (OU, GV, IC).

In ZP, ephemeral IDs change randomly each τ seconds and they are not exchanged through BLE, thus the attack cannot be performed.

These results are compliant with Theorem 6.11 that shows that LD and UD are vulnerable to C_1 in (OU, GV, IC), while ZP is not vulnerable to C_1 in (OU, GV, IC).

We highlight that, although the attacks regard DP-3T, they also apply to other decentralized protocols [55, 145, 50] as the vulnerabilities are due to the exchange of identifiers.

The vulnerabilities of DP-3T and ZE2-P3T to the attacks are summarized in Table 6.2.



Attack	LD	UD	ZP
Paparazzi	✗	✓	✓
Orwell	✗	✗	✓
Brutus	✗	✗	✓
Gossip	✗	✗	✓
Matteotti	✓	✗	✗
Missile	✗	✗	✓
Fregoli/Relay	✗	✗	✓
Battleship	✓	✓	✓
Little Thumb	✗	✗	✓

Table 6.2: Vulnerabilities of DP-3T and ZE2-P3T to the attacks. ✗ means vulnerable while ✓ means resistant.

6.8 Discussion

The COVID-19 pandemic certainly represents one of the most difficult challenges that modern society has ever faced. Moreover, history teaches us that fighting a pandemic has always been a very difficult task and, despite the scientific and technological advances our society enjoys today, we have experienced that a pandemic is still capable of threatening and seriously damaging the world, and its social and economic systems.

The strategy to adopt to fight a pandemic strongly depends on the characteristic of the pathogen and the disease at the basis of the pandemic. But, in general, the strategy is a combination of different measures.

Contact tracing (both traditional and digital) is one of the weapons in the heads of governments, but it is not enough. It should be suitably combined with containment measures. A possible example of containment measures are the following: health checkpoints for passengers, mandatory supervised quarantine, mandatory communication, red zones, banning air traffic from specific areas of the world, vaccination, suspensions of public events, general lockdown or movement restrictions, suspension of all commercial activities non-indispensable for production, an extension of the ban on non-indispensable activities, etc. Contact tracing can be effective in some phases of the pandemic, and could be also restricted to specific areas of the country in which the conditions are appropriate. Specifically, contact tracing can be fruitful if the incidence of the infection is below a given threshold. Indeed, the role of contact tracing is to enable early reactions by applying additional strict containment measures (as quarantine) only to those people that are identified by the tracing activity. Conversely, when the incidence of the pathology is high, the role of contact tracing decreases and also its feasibility, due to the number of cases to manage. On the other hand, in this case, a general containment protocol can be applied. All the above strategies can encounter a number of obstacles, that are cultural, social, technological, economic, etc. The obstacles themselves may become opportunities for designing more effective strategies also by reducing their negative impact. For example, heterogeneity of the population introduces specific issues to consider. Certain differences in intrinsic characteristics of the population may influence the dynamics of the local epidemic. Therefore, governments should consider local conditions to adopt heterogeneous containment policies. Furthermore, another factor that can hinder the management of a pandemic, especially when the symptoms may be similar, is the coexistence with other epidemics (like the seasonal flu).

In this chapter, we focus on digital contact tracing, by proposing a new protocol that improves the state of the art from the perspective of security and privacy. The motivation of this chapter is that the prerequisite for the effectiveness of digital contact tracing is above all that it should be accepted by the population, in addition to the fact that it should be adequately combined with containment measures. Therefore, security and privacy features take a central role.

Concerning security, in particular, as some attacks that we contrast (like the relay attack – also said Fregoli attack) can threaten the stability of large communities, governments may also be seriously interested in this aspect.

Consider that the relay attack could be performed on a massive scale, potentially quarantining many people. For example, this could be exploited in the case of elections, to inhibit their regular running. Therefore, the presented research has a specific focus on security and privacy aspects, without neglecting the feasibility of the proposed solution, which is a relevant point, considering that it should be designed for a huge number of users. A final aspect to discuss is if the approach presented in this paper is generalizable to all contagious diseases. Obviously, the answer to this question should take into account the specific characteristics of the considered disease. Apart from the obvious fact that even short-time and non-close contacts could be meaningful only for respiratory infections, a contact tracing solution is suitable for a specific disease in function of the safety distance, the role of surfaces and share objects, the incubation period, the speed of the diagnostic test, the presence of paucisymptomatic or asymptomatic cases, and so on. The number of variables to consider are so high that, as the recent scientific literature witnesses for the case of COVID-19, it is preferable to restrict the action range to a single disease, which is the choice adopted in this chapter.

6.9 Future Works

We proposed a novel contact tracing protocol that improves the state of the art from the perspective of security and privacy. As future work, we plan to transform the software prototype presented in this chapter into a complete software system, to run it in real-life contexts. Indeed, we argue that a real-life experimentation possibly with the partnership of industrial and government parties would be the desirable follow-up of the present scientific study. Finally, we plan to embed into the protocol the optional optimization features included in the implementation to carefully study their security implications.

7

Conclusion

In this thesis, we proposed new approaches to realize proximity-based services while guaranteeing privacy. The problem we addressed is that proximity-based services expose the user to serious privacy threats because they could allow massive monitoring by an honest but curious provider.

We presented several protocols to implement proximity-based services in different contexts, such as social networks, electronic proximity monitoring for the prevention of crimes, and contact tracing, always with the main objective of ensuring privacy.

After analyzing the existing literature related to this field, first of all, we focused on the management of the mapping of the territory. Therefore, we developed an efficient representation capable of supporting proximity detection at different ranges. To do this, we have proposed a tag-grid-based approach. The grid-based techniques consist of the partition of the territory into *cells* of a certain shape (squares, hexagons, circles, etc.), possibly overlapping each other. Since we wanted to enable the modulation of the size of the searching area in which users want to perform their proximity test, we needed a more sophisticated structure. Therefore, we designed a new hierarchical spatial index, called *shifted quad tree* allowing the user to choose the distance within which proximity testing is performed. We combined our grid-based approach with a *tag-based* mechanism.

Then, we tried to solve the data integrity problem, since the proximity service provider can outsource the map data to a third party (typically the cloud), which however may not be honest. The proximity service provider needs guarantees on the completeness and correctness of the portion of map data returned. To solve this problem, we have developed a new technique to guarantee query integrity over map data outsourced to a cloud. The proposed technique outperforms tree-like state-of-the-art solutions and shows the nice feature of providing guarantees for freshness without requiring timestamps, synchronization, and revocation mechanisms. We only considered spatial range queries, and only evaluates the proposed method through an analytic approach. In favor of fairness, this is done by considering lower bound (Ω) costs for tree-like competitor techniques, not referring to a specific technique, thus without considering that the effective implementation of insertions and deletions in any existing tree-like method is not natural to the point that the scientific literature considers these methods not well-applicable in the dynamic case.

The first context we have considered is that of social networks. By adopting the structure we created and after defining an anonymity protocol and communication primitives, we implemented a proximity testing protocol protecting users' privacy against the global adversary. We developed a privacy-preserving proximity-based solution that provides both symmetric and asymmetric proximity testing entirely within social networks. In particular, we realized three different proximity-based services. The first is a service aimed to test the proximity of users who know each other. Roughly, we provided the privacy features to a service similar to *Facebook Nearby Friends*. The second service is used to test the proximity between users who do not know each other but make public some information. The service allows detecting proximity of unknown users only on the basis of their agreement. This service extends the features given by services such as *Tinder*, by enabling the above privacy features. Finally, the last service regards proximity testing of a user with respect to a (static or moving) target. The privacy requirement is that the user remains anonymous also with respect to the target. This service extends services such as *Tripadvisor* or *BlaBlaCar*. We call this service `TN-service` (standing for *target nearby service*). We provided the security analysis of our solution.

The second context considered concerns Electronic Proximity Monitoring for the Prevention of Crimes. Existing solutions (RFID and GPS) suffer from security and privacy issues. We proposed a more complex solution involving a hierarchical grid-based approach and the collaboration of a telephone service provider. We designed a privacy-preserving GPS-based solution that does not allow the victim's location to be revealed unless the offender is nearby. Our solution advances the state of the art. We provided the security analysis of our solution.

The last context considered is that of contact tracing. We proposed a centralized digital contact tracing (DCT) protocol, called ZE2-P3T (Zero Ephemeral Exchanging Privacy-Preserving Proximity Protocol), which relies on smartphone localization but does not give any information about the user's location and identity to the server. The most important feature of our approach is that such ephemeral identifiers are not exchanged among users, as it happens for the state-of-the-art decentralized protocol DP-3T/GAEN. This is important because this exchange is the basis of most vulnerabilities of DP-3T/GAEN. As a consequence, the proposed approach is definitely more secure than DP-3T/GAEN. This fact represents the main result of our solution. To show this, we defined a theoretical framework able to quantitatively compare the two approaches, according to a suitably weighted ordering relation. Interestingly, our protocol is not based on BLE. Therefore, the users are not forced to turn on the Bluetooth interface (as it happens for DP-3T/GAEN). This shields smartphones from other attacks that exploit Bluetooth. An important point to remark is that the proposed solution has been also analyzed experimentally, to test its feasibility and also to provide a methodology to follow in a real-life context to size the server-side resources. We developed a software prototype that implements the main functionalities of the solution.

In conclusion, the solutions proposed in this thesis resolve privacy and security issues and advance the state of the art.

As future work, we plan to apply the technique for data integrity not only for cloud-based data outsourcing but also in the field of blockchain inter-ledger protocols. Furthermore, the proposed approach to provide proximity-based services is a plausible example of how to achieve privacy objectives while keeping the current centralized social network architectures therefore we plan to implement a real-life application in the future. Regarding the proposed protocol for electronic monitoring preserving the privacy of the victim, we program to implement the solution and experiment with it in a real-life environment. Finally, we plan to transform the software prototype of our digital contact tracing protocol into a complete software system, to run it in real-life contexts. Indeed, we argue that a real-life experimentation possibly with the partnership of industrial and government parties would be the desirable follow-up of our scientific study.

References

1. Aarogya setu app, india. `https://www.mygov.in/aarogya-setu-app`, 2020.
2. Apturicovid, republic of latvia. `https://github.com/ApturiCOVID/apturicovid-ios`, 2020.
3. Covid-safepaths app, united states. `https://github.com/Path-Check/covid-safe-paths`, 2020.
4. Covid watch app, usa. `https://www.covidwatch.org`, 2020.
5. Covidsafe app, australia. `https://github.com/AU-COVIDSafe/`, 2020.
6. Hamagen app. `https://github.com/MohGovIL/hamagen-react-native`, 2020.
7. Immuni app, italy. `https://github.com/immuni-app`, 2020.
8. Privacy and security risk evaluation of digital proximity tracing systems. `https://github.com/DP-3T/documents/blob/master/Securityanalysis/PrivacyandSecurityAttacksonDigitalProximityTracingSystems.pdf`, 2020.
9. Stopcovid app, france. `https://gitlab.inria.fr/stopcovid1`, 2020.
10. Stoppcorona app, austria. `https://github.com/austrianredcross/`, 2020.
11. Swisscovid app, switzerland. `https://github.com/DP-3T/dp3t-app-android-ch`, 2020.
12. Tracetogether app, singapore. `https://github.com/jontejj/tracetogether`, 2020.
13. Electronic monitoring smartphone apps: An analysis of risks from technical, Human-Centered, and legal perspectives. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, August 2022. USENIX Association.
14. Nadeem Ahmed, Regio A Michelin, Wanli Xue, Sushmita Ruj, Robert Malaney, Salil S Kanhere, Aruna Seneviratne, Wen Hu, Helge Janicke, and Sanjay K Jha. A survey of covid-19 contact tracing apps. *IEEE access*, 8:134577–134601, 2020.
15. Fraunhofer AISEC. Pandemic contact tracing apps: Dp-3t, pepp-pt ntk, and robert from a privacy perspective. *IACR Cryptol. ePrint Arch.*, 2020:489, 2020.
16. Seham A Alansari, Mahmoud M Badr, Mohamed Mahmoud, and Waleed Alasmary. Efficient and privacy-preserving contact tracing system for covid-19 using blockchain. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2021.
17. Hans-Joerg Albrecht. Electronic monitoring in europe. *Bulletin d information penologique19*, 20:8–9, 1995.

18. Thamer Altuwaiyan, Mohammad Hadian, and Xiaohui Liang. Epic: efficient privacy-preserving contact tracing for infection detection. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.
19. Dario Ortega Anderez, Eiman Kanjo, Amna Amnwar, Shane Johnson, and David Lucy. The rise of technology in crime prevention: Opportunities, challenges and practitioners perspectives. *arXiv preprint arXiv:2102.04204*, 2021.
20. Apple and Google. Apple and google's exposure notification system, 2020.
21. Claudio A Ardagna, Marco Cremonini, Ernesto Damiani, S De Capitani di Vimercati, and Pierangela Samarati. Location privacy protection through obfuscation-based techniques. In *IFIP annual conference on data and applications security and privacy*, pages 47–60. Springer, 2007.
22. Benedikt Auerbach, Suvradip Chakraborty, Karen Klein, Guillermo Pascual-Perez, Krzysztof Pietrzak, Michael Walter, and Michelle Yeo. Inverse-sybil attacks in automated contact tracing. In *Cryptographers' Track at the RSA Conference*, pages 399–421. Springer, 2021.
23. Gennaro Avitabile, Vincenzo Botta, Vincenzo Iovino, and Ivan Visconti. Towards defeating mass surveillance and sars-cov-2: The pronto-c2 fully decentralized automatic contact tracing system. *Cryptology ePrint Archive*, 2020.
24. Gennaro Avitabile, Daniele Friolo, and Ivan Visconti. Terrorist attacks for fake exposure notifications in contact tracing systems. In *International Conference on Applied Cryptography and Network Security*, pages 220–247. Springer, 2021.
25. Ahmed Barnawi, Prateek Chhikara, Rajkumar Tekchandani, Neeraj Kumar, and Bander Alzahrani. Artificial intelligence-enabled internet of things-based system for covid-19 screening using aerial thermal imaging. *Future Generation Computer Systems*, 124:119–132, 2021.
26. Lars Baumgärtner, Alexandra Dmitrienko, Bernd Freisleben, Alexander Gruler, Jonas Höchst, Joshua Kühlberg, Mira Mezini, Richard Mitev, Markus Miettinen, Anel Muhamedagic, et al. Mind the gap: Security & privacy risks of contact tracing apps. In *2020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom)*, pages 458–467. IEEE, 2020.
27. Jason Bay, Joel Kek, Alvin Tan, Chai Sheng Hau, Lai Yongquan, Janice Tan, and Tang Anh Quy. Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders. *Government Technology Agency-Singapore, Tech. Rep*, 18, 2020.
28. Amos Beimel and Shlomi Dolev. Buses for anonymous message delivery. *Journal of Cryptology*, 16(1), 2003.
29. Jyoti Belur, Amy Thornton, Lisa Tompson, Matthew Manning, Aiden Sidebottom, and Kate Bowers. A systematic review of the effectiveness of the electronic monitoring of offenders. *Journal of Criminal Justice*, 68:101686, 2020.
30. Yoshua Bengio, Daphne Ippolito, Richard Janda, Max Jarvie, Benjamin Prud'homme, Jean-François Rousseau, Abhinav Sharma, and Yun William Yu. Inherent privacy limitations of decentralized contact tracing apps. *Journal of the American Medical Informatics Association*, 28(1):193–195, 2021.

31. Krista Bennett and Christian Grothoff. Gap–practical anonymous networking. In *International Workshop on Privacy Enhancing Technologies*, pages 141–160. Springer, 2003.
32. Alastair R Beresford and Frank Stajano. Mix zones: User privacy in location-aware services. In *IEEE Annual conference on pervasive computing and communications workshops, 2004. Proceedings of the Second*, pages 127–131. IEEE, 2004.
33. Wasilij Beskorovajnov, Felix Dörre, Gunnar Hartung, Alexander Koch, Jörn Müller-Quade, and Thorsten Strufe. Contra corona: Contact tracing against the coronavirus by bridging the centralized–decentralized divide for stronger privacy. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 665–695. Springer, 2021.
34. Behnaz Bostanipour and Benoît Garbinato. Effective and efficient neighbor detection for proximity-based mobile applications. *Computer Networks*, 79:216–235, 2015.
35. Andre Bourdoux, Andre Noll Barreto, Barend van Liempd, Carlos de Lima, Davide Dardari, Didier Belot, Elana-Simona Lohan, Gonzalo Seco-Granados, Hadi Sarieddeen, Henk Wymeersch, et al. 6g white paper on localization and sensing. *arXiv preprint arXiv:2006.01779*, 2020.
36. Samuel Brack, Leonie Reichert, and Björn Scheuermann. Caudht: decentralized contact tracing using a dht and blind signatures. In *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, pages 337–340. IEEE, 2020.
37. Thomas Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.
38. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, and Cecilia Labrini. Anonymous short communications over social networks. In *International Conference on Security and Privacy in Communication Systems*, pages 43–63. Springer, 2021.
39. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, and Cecilia Labrini. A privacy-preserving protocol for proximity-based services in social networks. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021.
40. Francesco Buccafurri, Vincenzo De Angelis, Maria Francesca Idone, and Cecilia Labrini. A protocol for anonymous short communications in social networks and its application to proximity-based services. *Online Social Networks and Media*, 31:100221, 2022.
41. Francesco Buccafurri, Vincenzo De Angelis, and Cecilia Labrini. A privacy-preserving solution for proximity tracing avoiding identifier exchanging. In *2020 International Conference on Cyberworlds (CW)*, pages 235–242. IEEE, 2020.
42. Francesco Buccafurri, Vincenzo De Angelis, and Cecilia Labrini. Integrity guarantees over highly dynamic outsourced map data. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1691–1696, 2021.
43. Francesco Buccafurri, Vincenzo De Angelis, and Cecilia Labrini. A centralized contact-tracing protocol for the covid-19 pandemic. *Information Sciences*, 2022.
44. Francesco Buccafurri, Filippo Furfaro, Giuseppe M Mazzeo, and Domenico Saccà. A quad-tree based multiresolution approach for two-dimensional summary data. *Information Systems*, 36(7):1082–1103, 2011.

45. Philip Bulman. Electronic monitoring reduces recidivism. *Corrections Today*, 72(6), 2010.
46. Deeanna M Button, Matthew DeMichele, and Brian K Payne. Using electronic monitoring to supervise sex offenders: Legislative patterns and implications for community corrections officers. *Criminal Justice Policy Review*, 20(4):414–436, 2009.
47. Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. Robert: Robust and privacy-preserving proximity tracing. 2020.
48. Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. Desire: A third way for a european exposure notification system leveraging the best of centralized and decentralized systems. *arXiv preprint arXiv:2008.01621*, 2020.
49. Zdenek Chaloupka. Technology and standardization gaps for high accuracy positioning in 5g. *IEEE Communications Standards Magazine*, 1(1):59–65, 2017.
50. Justin Chan, Dean Foster, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Puneet Sharma, et al. Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing. *arXiv preprint arXiv:2004.03544*, 2020.
51. Path Check. Project aurora: A new open source solution for the google apple exposure notification api. 2020.
52. Bo-Rong Chen and Yih-Chun Hu. Mitigating denial-of-service attacks on digital contact tracing. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 770–771, 2020.
53. Weiwei Cheng, HweeHwa Pang, and Kian-Lee Tan. Authenticating multi-dimensional query results in data publishing. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 60–73. Springer, 2006.
54. Hyunghoon Cho, Daphne Ippolito, and Yun William Yu. Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs. *arXiv preprint arXiv:2003.11511*, 2020.
55. TCN Coalition. Tcn protocol for decentralized, privacy-preserving contact tracing, 2020.
56. Aaqib Bashir Dar, Auqib Hamid Lone, Saniya Zahoor, Afshan Amin Khan, and Roohie Naaz. Applicability of mobile contact tracing in fighting pandemic (covid-19): Issues, challenges and solutions. *Computer Science Review*, 38:100307, 2020.
57. Guido De Angelis, Valter Pasku, Alessio De Angelis, Marco Dionigi, Mauro Mongiardo, Antonio Moschitta, and Paolo Carbone. An indoor ac magnetic positioning system. *IEEE Transactions on Instrumentation and Measurement*, 64(5):1267–1275, 2014.
58. Matthew Demichele, Brian K Payne, and Deeanna M Button. Electronic monitoring of sex offenders: Identifying unanticipated consequences and implications. *Journal of Offender Rehabilitation*, 46(3-4):119–135, 2008.
59. Ensheng Dong, Hongru Du, and Lauren Gardner. An interactive web-based dashboard to track covid-19 in real time. *The Lancet infectious diseases*, 20(5):533–534, 2020.

60. EU eHealth Network. Mobile applications to support contact tracing in the eu's fight against covid-19: Common eu toolbox for member states. *June-2020*, 2020.
61. Kathy Eivazi. Computer use monitoring and privacy at work. *Computer Law & Security Review*, 27(5):516–523, 2011.
62. Bin Fan, Dave G Andersen, Michael Kaminsky, and Michael D Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proc. of the 10th ACM International Conference on emerging Networking Experiments and Technologies*, pages 75–88, 2014.
63. Henrique Faria, Sara Paiva, and Pedro Pinto. An advertising overflow attack against android exposure notification system impacting covid-19 contact tracing applications. *IEEE Access*, 9:103365–103375, 2021.
64. Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. *Science*, 368(6491), 2020.
65. Mahdi Daghmehchi Firoozjaei, Jaegwan Yu, Hyoungkee Choi, and Hyoungshick Kim. Privacy-preserving nearest neighbor queries using geographical features of cellular networks. *Computer Communications*, 98:11–19, 2017.
66. Flora Fitzalan Howard. The experience of electronic monitoring and the implications for effective use. *The Howard Journal of Crime and Justice*, 59(1):17–43, 2020.
67. Official Exposure Notification App for Germany. Corona-warn-app. 2020.
68. Michael J Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 193–206, 2002.
69. Lalit Garg, Emeka Chukwu, Nidal Nasser, Chinmay Chakraborty, and Gaurav Garg. Anonymity preserving iot-based covid-19 and other infectious disease contact tracing model. *Ieee Access*, 8:159402–159414, 2020.
70. Bugra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *IEEE Transactions on Mobile Computing*, 7(1):1–18, 2007.
71. Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: anonymizers are not necessary. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 121–132, 2008.
72. Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42, 2003.
73. Per Hallgren, Martin Ochoa, and Andrei Sabelfeld. Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6. IEEE, 2015.
74. Rita Haverkamp, Markus Mayer, and Rene Levy. Electronic monitoring in europe. *Eur. J. Crime Crim. L. & Crim. Just.*, 12:36, 2004.
75. Yan He and Jiageng Chen. User location privacy protection mechanism for location-based services. *Digital communications and networks*, 7(2):264–276, 2021.

76. Urs Hengartner. Enhancing user privacy in location-based services. *Centre for Applied Cryptographic Research, University of Waterloo, Tech. Rep. CACR*, 27, 2006.
77. Urs Hengartner. Hiding location information from location-based services. In *2007 International Conference on Mobile Data Management*, pages 268–272. IEEE, 2007.
78. Andreas Hirt, Michael Jacobson, and Carey Williamson. Taxis: scalable strong anonymous communication. In *2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*, pages 1–10. IEEE, 2008.
79. Ling Hu, Wei-Shinn Ku, Spiridon Bakiras, and Cyrus Shahabi. Spatial query integrity with voronoi neighbors. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):863–876, 2011.
80. Peizhao Hu, Tamalika Mukherjee, Alagu Valliappan, and Stanislaw Radziszowski. Homomorphic proximity computation in geosocial networks. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 616–621. IEEE, 2016.
81. Peng Hu. Iot-based contact tracing systems for infectious diseases: Architecture and analysis. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–7. IEEE, 2020.
82. Cheng Huang, Rongxing Lu, Hui Zhu, Jun Shao, Abdulrahman Alamer, and Xiaodong Lin. Eppd: efficient and privacy-preserving proximity testing with differential privacy techniques. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2016.
83. Harriet Hunt-Grubbe. The many faces of surveillance: Ethical considerations that encompass the use of electronic monitoring in criminal and clinical populations. In *Ethical Issues in Clinical Forensic Psychiatry*, pages 115–134. Springer, 2020.
84. Jarkko Hyysalo, Sandun Dasanayake, Jari Hannu, Christian Schuss, Mikko Rajanen, Teemu Leppänen, David Doermann, and Jaakko Sauvola. Smart mask–wearable iot solution for improved protection and personal health. *Internet of Things*, 18:100511, 2022.
85. Kimmo Järvinen, Ágnes Kiss, Thomas Schneider, Oleksandr Tkachenko, and Zheng Yang. Faster privacy-preserving location proximity schemes. In *International Conference on Cryptology and Network Security*, pages 3–22. Springer, 2018.
86. Hongbo Jiang, Jie Li, Ping Zhao, Fanzi Zeng, Zhu Xiao, and Arun Iyengar. Location privacy-preserving mechanisms in location-based services: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(1):1–36, 2021.
87. Ting Jiang, Yang Zhang, Minhao Zhang, Ting Yu, Yizheng Chen, Chenhao Lu, Ji Zhang, Zhao Li, Jun Gao, and Shuigeng Zhou. A survey on contact tracing: the latest advancements and challenges. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 8(2):1–35, 2022.
88. Mouna Kacimi, Stefano Ortolani, and Bruno Crispo. Anonymous opinion exchange over untrusted social networks. In *Proceedings of the second ACM EuroSys workshop on social network systems*, pages 26–32, 2009.
89. Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE transactions on knowledge and data engineering*, 19(12):1719–1733, 2007.

90. Günter Kampf, Daniel Todt, Stephanie Pfaender, and Eike Steinmann. Persistence of coronaviruses on inanimate surfaces and their inactivation with biocidal agents. *Journal of hospital infection*, 104(3):246–251, 2020.

91. Ali Khoshgozaran, Houtan Shirani-Mehr, and Cyrus Shahabi. Spiral: A scalable private information retrieval approach to location privacy. In *2008 Ninth International Conference on Mobile Data Management Workshops, MDMW*, pages 55–62. IEEE, 2008.

92. James Kilgore. Progress or more of the same? electronic monitoring and parole in the age of mass incarceration. *Critical Criminology*, 21(1):123–139, 2013.

93. Panayiotis Kotzanikolaou, George Chatzisofroniou, and Mike Burmester. Broadcast anonymous routing (bar): scalable real-time anonymous communication. *International Journal of Information Security*, 16(3):313–326, 2017.

94. Panayiotis Kotzanikolaou, Constantinos Patsakis, Emmanouil Magkos, and Michalis Korakakis. Lightweight private proximity testing for geospatial social networks. *Computer Communications*, 73:263–270, 2016.

95. Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Hmac: Keyed-hashing for message authentication. Technical report, 1997.

96. Douglas J Leith and Stephen Farrell. Coronavirus contact tracing: Evaluating the potential of using bluetooth received signal strength for proximity detection, 2020.

97. Douglas J Leith and Stephen Farrell. Contact tracing app privacy: What data is shared by europe's gaen contact tracing apps. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

98. Hong Ping Li, Haibo Hu, and Jianliang Xu. Nearby friend alert: Location anonymity in mobile geosocial networks. *IEEE Pervasive Computing*, 12(4):62–70, 2012.

99. Jinfeng Li and Xinyi Guo. Global deployment mappings and challenges of contact-tracing apps for covid-19. *Available at SSRN 3609516*, 2020.

100. Mingyan Li, Iordanis Koutsopoulos, and Radha Poovendran. Optimal jamming attack strategies and network defense policies in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(8):1119–1133, 2010.

101. Xianghong Li, Dongyan Wei, Qifeng Lai, Ying Xu, and Hong Yuan. Smartphone-based integrated pdr/gps/bluetooth pedestrian location. *Advances in Space Research*, 59(3):877–887, 2017.

102. Chang Liu, Rajiv Ranjan, Chi Yang, Xuyun Zhang, Lizhe Wang, and Jinjun Chen. Murdpa: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud. *IEEE Transactions on Computers*, 64(9):2609–2622, 2014.

103. Kun Liu, Chris Giannella, and Hillol Kargupta. An attacker's view of distance preserving maps for privacy preserving data mining. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 297–308. Springer, 2006.

104. Ying Liu, Xiufang Shi, Shibo He, and Zhiguo Shi. Prospective positioning architecture and technologies in 5g networks. *IEEE Network*, 31(6):115–121, 2017.

105. G Lockhart-Mirams, C Pickles, and E Crowhurst. Cutting crime: the role of tagging in offender management. *London: Reform*, 2015.

106. J Lyle and Ani Sunny. Spatial query authentication using merkle quadtree. *International Journal of Engineering Trends and Technology*, 59:186–191, 05 2018.
107. Halgurd S Maghdid, Ihsan Alshahib Lami, Kayhan Zrar Ghafoor, and Jaime Lloret. Seamless outdoors-indoors localization solutions on smartphones: implementation and challenges. *ACM Computing Surveys (CSUR)*, 48(4):1–34, 2016.
108. Magdalena Putka Marzena Kordaczuk-Was. Integrated system of monitoring cases of domestic violence - spanish experience, 2011/2012.
109. Sergio Mascetti, Letizia Bertolaja, and Claudio Bettini. A practical location privacy attack in proximity services. In *2013 IEEE 14th International Conference on Mobile Data Management*, volume 1, pages 87–96. IEEE, 2013.
110. Sergio Mascetti, Claudio Bettini, Dario Freni, and X Sean Wang. Spatial generalisation algorithms for lbs privacy preservation. *Journal of Location Based Services*, 1(3):179–207, 2007.
111. Sergio Mascetti, Claudio Bettini, Dario Freni, X Sean Wang, and Sushil Jajodia. Privacy-aware proximity based services. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 31–40. IEEE, 2009.
112. Sergio Mascetti, Dario Freni, Claudio Bettini, X Sean Wang, and Sushil Jajodia. Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. *The VLDB journal*, 20(4):541–566, 2011.
113. Ralph C Merkle. A certified digital signature. In *Conference on the Theory and Application of Cryptology*, pages 218–238. Springer, 1989.
114. Ralph C Merkle. One way hash functions and des. In *Conference on the Theory and Application of Cryptology*, pages 428–446. Springer, 1990.
115. Ralph Charles Merkle. *Secrecy, authentication, and public key systems.* Stanford university, 1979.
116. Microprocessor, MS Committee, et al. Ieee standard specifications for public-key cryptography. *IEEE Computer Society*, pages 1–226, 2000.
117. Ehsan Moghadas, Javad Rezazadeh, and Reza Farahbakhsh. An iot patient monitoring based on fog computing and data mining: Cardiac arrhythmia usecase. *Internet of Things*, 11:100251, 2020.
118. Rajani Muraleedharan and Lisa Ann Osadciw. Jamming attack detection and countermeasures in wireless sensor network using ant system. In *Wireless Sensing and Processing*, volume 6248, page 62480G. International Society for Optics and Photonics, 2006.
119. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43, 1989.
120. Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, Dan Boneh, et al. Location privacy via private proximity testing. In *NDSS*, volume 11, 2011.
121. Mike Nellis. The electronic monitoring of offenders in england and wales: Recent developments and future prospects. *The British Journal of Criminology*, 31(2):165–185, 1991.
122. Mike Nellis. Electronic monitoring in scotland 1998-2006. *Scottish Journal of Criminal Justice Studies*, 12:74–96, 2006.

123. Mike Nellis. Understanding the electronic monitoring of offenders in europe: Expansion, regulation and prospects. *Crime, Law and Social Change*, 62(4):489–510, 2014.

124. Mike Nellis. Electronic monitoring around the world, 04 2021.

125. Pai Chet Ng, James She, and Rong Ran. A compressive sensing approach to detect the proximity between smartphones and ble beacons. *IEEE Internet of Things Journal*, 6(4):7162–7174, 2019.

126. Mohammad Reza Nosouhi, Shui Yu, Keshav Sood, and Marthie Grobler. Hsdc–net: secure anonymous messaging in online social networks. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 350–357. IEEE, 2019.

127. International Association of Chiefs of Police. Tracking sex offenders with electronic monitoring technology: Implications and practical uses for law enforcement, 2008.

128. Ivan Oleynikov, Elena Pagnin, and Andrei Sabelfeld. Where are you bob? privacy-preserving proximity testing with a napping party. In *European Symposium on Research in Computer Security*, pages 677–697. Springer, 2020.

129. Liwei Ouyang, Yong Yuan, Yumeng Cao, and Fei-Yue Wang. A novel framework of collaborative early warning for covid-19 based on blockchain and smart contracts. *Information sciences*, 570:124–143, 2021.

130. Kathy G Padgett, William D Bales, and Thomas G Blomberg. Under surveillance: An empirical test of the effectiveness and consequences of electronic monitoring. *Criminology & Public Policy*, 5(1):61–91, 2006.

131. Antonio Iyda Paganelli, Pedro Elkind Velmovitsky, Pedro Miranda, Adriano Branco, Paulo Alencar, Donald Cowan, Markus Endler, and Plinio Pelegrini Morita. A conceptual iot-based early-warning architecture for remote monitoring of covid-19 patients in wards and at home. *Internet of Things*, 18:100399, 2022.

132. HweeHwa Pang, Arpit Jain, Krithi Ramamritham, and Kian-Lee Tan. Verifying completeness of relational query results in data publishing. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 407–418, 2005.

133. Dimitrios Papadopoulos, Stavros Papadopoulos, and Nikos Triandopoulos. Taking authenticated range queries to arbitrary dimensions. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 819–830, 2014.

134. Zhe Peng, Cheng Xu, Haixin Wang, Jinbin Huang, Jianliang Xu, and Xiaowen Chu. P2b-trace: Privacy-preserving blockchain-based contact tracing to combat pandemics. In *Proceedings of the 2021 international conference on management of data*, pages 2389–2393, 2021.

135. Krzysztof Pietrzak. Delayed authentication: Preventing replay and relay attacks in private contact tracing. In *International Conference on Cryptology in India*, pages 3–15. Springer, 2020.

136. B PRENEEL, J VANDEWALLE, and Bart VAN ROMPAY. Analysis and design of cryptographic hash functions, mac algorithms and block ciphers. 2004.

137. Bart Preneel. *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit te Leuven Leuven, 1993.

138. Bart Preneel. Cryptographic hash functions. *European Transactions on Telecommunications*, 5(4):431–448, 1994.

139. Umair Muitaba Qureshi, Gerhard Petrus Hancke, Teklay Gebremichael, Ulf Jennehag, Stefan Forsström, and Mikael Gidlund. Survey of proximity based authentication mechanisms for the industrial internet of things. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pages 5246–5251. IEEE, 2018.

140. R Rajadurai, R Suganyaa, V Santhakumari, B Srilakshmi, K Prem Kumar, and S Vasantharaj. Proximity based security for location based services. In *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET 2015)*, pages 1–5, 2015.

141. Asslinah Mocktoolah Ramtohul and Kavi Kumar Khedo. Proximity based social networking in urban environments: Applications, architectures and frameworks. In *Precision Positioning with Commercial Smartphones in Urban Environments*, pages 71–107. Springer, 2021.

142. Leonie Reichert, Samuel Brack, and Björn Scheuermann. Privacy-preserving contact tracing of covid-19 patients. *Cryptology ePrint Archive*, 2020.

143. Marc Renzema and David Thomas Skelton. *Use of electronic monitoring in the United States: 1989 update*. US Department of Justice, Office of Justice Programs, 1990.

144. Kai Riemer, Raffaele Ciriello, Sandra Peter, and Daniel Schlagwein. Digital contact-tracing adoption in the covid-19 pandemic: It governance for collective action at the societal level. *European Journal of Information Systems*, 29(6):731–745, 2020.

145. Ronald L Rivest, Jon Callas, Ran Canetti, Kevin Esvelt, Daniel Kahn Gillmor, Yael Tauman Kalai, Anna Lysyanskaya, Adam Norige, Ramesh Raskar, Adi Shamir, et al. The pact protocol specification. private automated contact tracing team, 2020.

146. Peter Ruppel, Georg Treu, Axel Küpper, and Claudia Linnhoff-Popien. Anonymous user tracking for location-based community services. In *International Symposium on Location- and Context-Awareness*, pages 116–133. Springer, 2006.

147. Pierangela Samarati. Data security and privacy in the cloud. In *International Conference on Information Security Practice and Experience*, pages 28–41. Springer, 2014.

148. Dariusz Sarzała. Electronic monitoring in the polish legal system as a form of social readaptation. *American Historical*, page 1916, 2016.

149. Annesley K Schmidt. *The Use of Electronic Monitoring by Criminal Justice Agencies, 1988*. US Department of Justice, National Institute of Justice, 1988.

150. Ralph Schwitzgebel, Robert Schwitzgebel, Walter N Pahnke, and William Sprech Hurd. A program of research in behavioral electronics. *Behavioral Science*, 9(3):233–238, 1964.

151. Jaroslav Šeděnka and Paolo Gasti. Privacy-preserving distance computation and proximity testing on earth, done right. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 99–110, 2014.

152. Otto Seiskari. Pble contact tracing sniffer poc. `https://github.com/oseiskar/coronasniffer`, 2020.

153. Laurynas Šikšnys, Jeppe R Thomsen, Simonas Šaltenis, Man Lung Yiu, and Ove Andersen. A location privacy aware friend locator. In *International Symposium on Spatial and Temporal Databases*, pages 405–410. Springer, 2009.

154. Laurynas Šikšnys, Jeppe Rishede Thomsen, Simonas Šaltenis, and Man Lung Yiu. Private and flexible proximity detection in mobile social networks. In *2010 Eleventh International Conference on Mobile Data Management*, pages 75–84. IEEE, 2010.

155. Jagdeep Singh, Isaac Woungang, Sanjay Kumar Dhurandher, and Khuram Khalid. A jamming attack detection technique for opportunistic networks. *Internet of Things*, 17:100464, 2022.

156. Rajeev Sobti and Ganesan Geetha. Cryptographic hash functions: a review. *International Journal of Computer Science Issues (IJCSI)*, 9(2):461, 2012.

157. Junggab Son, Donghyun Kim, Md Zakirul Alam Bhuiyan, Rahman Tashakkori, Jungtaek Seo, and Dong Hoon Lee. Privacy enhanced location sharing for mobile online social networks. *IEEE Transactions on Sustainable Computing*, 5(2):279–290, 2018.

158. B Sowmiya, VS Abhijith, S Sudersan, R Sakthi Jaya Sundar, M Thangavel, and P Varalakshmi. A survey on security and privacy issues in contact tracing application of covid-19. *SN computer science*, 2(3):1–11, 2021.

159. B Risteska Stojkoska, I Nizetic Kosovic, and T Jagust. A survey of indoor localization techniques for smartphones. In *th International conference ICT innovations*, 2016.

160. Sasu Tarkoma, Christian Esteve Rothenberg, and Eemil Lagerspetz. Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys & Tutorials*, 14(1):131–155, 2011.

161. Steven R Taylor, Subramaniam Kandaswamy, Timothy Evans, and David Mahaffey. *Market-survey of Location-based Offender Tracking Technologies*. Johns Hopkins University, Applied Physics Laboratory, 2016.

162. PEPP-PT Team. Pan-european privacy-preserving proximity tracing, 2020.

163. PEPP-PT Team. Pan-european privacy-preserving proximity tracing need-to-know system. overview of pepp-pt ntk, 2020.

164. Pietro Tedeschi, Spiridon Bakiras, and Roberto Di Pietro. Iotrace: a flexible, efficient, and privacy-preserving iot-enabled architecture for contact tracing. *IEEE Communications Magazine*, 59(6):82–88, 2021.

165. Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. *arXiv preprint arXiv:2004.13293*, 2020.

166. Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, et al. Decentralized privacy-preserving proximity tracing. *arXiv preprint arXiv:2005.12273*, 2020.

167. Sergio Varela-Santos and Patricia Melin. A new approach for classifying coronavirus covid-19 based on its manifestation on chest x-rays using texture features and neural networks. *Information sciences*, 545:403–414, 2021.

168. Serge Vaudenay. Analysis of dp3t-between scylla and charybdis. Technical report, 2020.

169. Serge Vaudenay. Centralized or decentralized? 2020.

170. Serge Vaudenay and Martin Vuagnoux. Little thumb attack on swisscovid. 2020.

171. Florian Vogt, Bridget Haire, Linda Selvey, Anthea L Katelaris, and John Kaldor. Effectiveness evaluation of digital contact tracing for covid-19 in new south wales, australia. *The Lancet Public Health*, 7(3):e250–e258, 2022.

172. Luis Von Ahn, Andrew Bortz, and Nicholas J Hopper. K-anonymous message transmission. In *Proceedings of the 10th ACM conference on Computer and Communications Security*, pages 122–130, 2003.

173. Qian Wang, Cong Wang, Jin Li, Kui Ren, and Wenjing Lou. Enabling public verifiability and data dynamics for storage security in cloud computing. In *European symposium on research in computer security*, pages 355–370. Springer, 2009.

174. Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE transactions on parallel and distributed systems*, 22(5):847–859, 2010.

175. Rolf H Weber. Internet of things–new security and privacy challenges. *Computer law & security review*, 26(1):23–30, 2010.

176. Kate Weisburd. Sentenced to surveillance: Fourth amendment limits on electronic monitoring. *NCL Rev.*, 98:717, 2019.

177. M Wekesa, MM Muendo, and A Mikinyango. The use of electronic tracking and monitoring systems and the right to privacy.

178. Jenny Williams and Don Weatherburn. Can electronic monitoring reduce reoffending? *Review of Economics and Statistics*, 104(2):232–245, 2022.

179. Klaus Witrisal, Paul Meissner, Erik Leitinger, Yuan Shen, Carl Gustafson, Fredrik Tufvesson, Katsuyuki Haneda, Davide Dardari, Andreas F Molisch, Andrea Conti, et al. High-accuracy localization for assisted living: 5g systems will turn multipath channels from foe to friend. *IEEE Signal Processing Magazine*, 33(2):59–70, 2016.

180. Yin Yang, Stavros Papadopoulos, Dimitris Papadias, and George Kollios. Authenticated indexing for outsourced spatial databases. *The VLDB Journal*, 18(3):631–648, 2009.

181. Ayong Ye, Qiuling Chen, Li Xu, and Wei Wu. The flexible and privacy-preserving proximity detection in mobile social network. *Future Generation Computer Systems*, 79:271–283, 2018.

182. Man Lung Yiu, Christian S Jensen, Xuegang Huang, and Hua Lu. Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In *2008 IEEE 24th International Conference on Data Engineering*, pages 366–375. IEEE, 2008.

183. Adam L Young and Moti Yung. The drunk motorcyclist protocol for anonymous communication. In *2014 IEEE Conference on Communications and Network Security*, pages 157–165. IEEE, 2014.

184. Faheem Zafari, Ioannis Papapanagiotou, Michael Devetsikiotis, and Thomas J Hacker. Enhancing the accuracy of ibeacons for indoor proximity-based services. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2017.

185. Bo Zhang, Boxiang Dong, and Wendy Hui Wang. Integrity authentication for sql query evaluation on outsourced databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1601–1618, 2019.

186. Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. Integridb: Verifiable sql for outsourced databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1480–1491, 2015.

187. Qingji Zheng, Shouhuai Xu, and Giuseppe Ateniese. Efficient query integrity for outsourced dynamic databases. In *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, pages 71–82, 2012.

188. Yao Zheng, Ming Li, Wenjing Lou, and Y Thomas Hou. Sharp: Private proximity test and secure handshake with cheat-proof location tags. In *European Symposium on Research in Computer Security*, pages 361–378. Springer, 2012.

189. Yao Zheng, Ming Li, Wenjing Lou, and Y Thomas Hou. Location based handshake and private proximity test with location tags. *IEEE Transactions on Dependable and Secure Computing*, 14(4):406–419, 2015.

190. Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, lester and pierre: Three protocols for location privacy. In *International Workshop on Privacy Enhancing Technologies*, pages 62–76. Springer, 2007.

191. Hui Zhu, Fengwei Wang, Rongxing Lu, Fen Liu, Gang Fu, and Hui Li. Efficient and privacy-preserving proximity detection schemes for social applications. *IEEE Internet of Things Journal*, 5(4):2947–2957, 2017.

192. Gaoqiang Zhuo, Qi Jia, Linke Guo, Ming Li, and Yuguang Fang. Privacy-preserving verifiable proximity test for location-based services. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

Scientific publications of Cecilia Labrini

1. Buccafurri Francesco, Labrini Cecilia, and Musarella Lorenzo. Smart-contract based Access Control on Distributed Information in a Smart-City scenario. *Proceedings of the 3rd Distributed Ledger Technology Workshop Co-located with ITASEC 2020, Ancona, Italy, 4 February, 2020. CEUR Workshop Proceedings 2580, CEUR-WS.org, 2020.*
2. Buccafurri Francesco, De Angelis Vincenzo, and Labrini Cecilia. A Privacy-Preserving Solution for Proximity Tracing Avoiding Identifier Exchanging. In *Proceedings of the 19th International Conference of CyberWorlds (CW) 2020, CAEN, France, 29 Sept-1 Oct, 2020. IEEE.*
3. Buccafurri Francesco, De Angelis Vincenzo, Labrini Cecilia, Lax Gianluca, Musarella Lorenzo, Nardone Roberto, and Russo Antonia. Model-driven Engineering for Swarm-based Space Exploration Missions. *International Astronautical Congress (IAC 2020). Online, 12-14 Oct. 2020.*
4. Buccafurri Francesco, De Angelis Vincenzo, and Labrini Cecilia. "Integrity guarantees over highly dynamic outsourced map data". In *Proceedings of the 36th Annual ACM Symposium on Applied Computing (pp. 1691-1696), 2021.*
5. Buccafurri Francesco, De Angelis Vincenzo, Idone Maria Francesca, and Labrini Cecilia. "WIP: An Onion-Based Routing Protocol Strengthening Anonymity". In *Proceedings of the 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM, Virtual Conference (pp. 231-235). IEEE, 2021*
6. Buccafurri Francesco, De Angelis Vincenzo, Idone Maria Francesca, and Labrini Cecilia. "A Distributed Location Trusted Service Achieving k-Anonymity against the Global Adversary". In *Proceedings of the 2021 22nd IEEE International Conference on Mobile Data Management, MDM, Virtual Conference (pp. 133-138). IEEE, 2021.*

7. Buccafurri Francesco, De Angelis Vincenzo, Idone Maria Francesca, and Labrini Cecilia. "Enabling Location k-Anonymity in Social Networks". In *Securing Social Networks in Cyberspace (pp. 35-50). CRC Press.*
8. Buccafurri Francesco, Consoli Angelo, Labrini Cecilia, and Mariotti Nesurini Alice. "A Solution to Support Integrity in the Lawful Interception Ecosystem". In *Proceedings of the International Conference on Electronic Government and the Information Systems Perspective, EGOVIS, (pp. 21-33), in the 32nd DEXA Conferences and Workshops, DEXA 2021. Springer, Cham, 2021.*
9. Buccafurri Francesco, De Angelis Vincenzo, Idone Maria Francesca, and Labrini Cecilia. "Extending Routes in Tor to Achieve Recipient Anonymity against the Global Adversary". In *Proceedings of the 2021 International Conference on Cyberworlds, CW 2021, Virtual Conference. IEEE, 2021.*
10. Buccafurri Francesco, De Angelis Vincenzo, Idone Maria Francesca, and Labrini Cecilia. "Anonymous Short Communications over Social Networks". In *Proceedings of the International Conference on Security and Privacy in Communication Systems, SECURECOMM 2021, Virtual Conference. Springer, Cham, 2021.*
11. Buccafurri Francesco, De Angelis Vincenzo, Idone Maria Francesca, and Labrini Cecilia. "Hardening Trust Models against Slandering Attacks in Relayed Content Delivery Services". In *Proceedings of the 2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2021, Virtual Conference. IEEE. 2021, October.*
12. Buccafurri Francesco, De Angelis Vincenzo, Idone Maria Francesca, and Labrini Cecilia. "A Privacy-Preserving Protocol for Proximity - Based Services in Social Networks". In *Proceedings of the 2021 International Conference on Global Communications Conference, GLOBECOM 2021, Virtual Conference. IEEE, 2021.*
13. Buccafurri Francesco, De Angelis Vincenzo, Idone Maria Francesca, Labrini Cecilia, and Lazzaro Sara. "Achieving Sender Anonymity in Tor against the Global Passive Adversary". *Applied Sciences, 12(1), 137. 2021.*
14. Buccafurri Francesco, De Angelis Vincenzo, Idone Maria Francesca, and Labrini Cecilia. "A protocol for anonymous short communications in social networks and its application to proximity-based services". *Online Social Networks and Media, 31, 100221, 2022.*
15. Buccafurri, Francesco, De Angelis, Vincenzo, and Labrini, Cecilia. "A Centralized Contact-Tracing Protocol for the COVID-19 Pandemic". *Information Sciences, 617, 103–132, Elsevier, 2022.*