POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

A Fully-Homomorphic Encryption System for Privacy-Preserving Network-Based Contact Tracing

Tesi di Laurea Magistrale in Telecommunication Engineering - Ingegneria delle Telecommunicazioni

Author: **Paul Horvath-Bojan**

Student ID: 942383
Advisor: Prof. Massimo Tornatore
Co-advisors: Dr. Davide Andreoletti, Dr. Omran Ayoub
Academic Year: 2022–23

Abstract

The Coronavirus outbreak from late 2019–early 2020 came with a number of challenges for humanity. Contact tracing is an important measure to decrease the further spread of the contagion. Traditionally, contact tracing systems are based on phone applications that check proximity with individuals confirmed positive for the virus and alert users of potential risks. The adoption of app-based solutions has been scarce, limiting their effectiveness. The main reasons for low adoption rates are either the majority of the population not being technologically savvy or not willing to have the apps installed on their phones. Additionally, the apps were perceived as posing a privacy risk, further hindering adoption. To overcome these issues, we present a network-based approach to contact tracing that relies on the emergence of 5G network coverage and geo-localization technologies which promise to reach sub-metre accuracy in upcoming years. The proposed system involves the cooperation between Mobile Operators (MO) and Government Agencies (GA). The former is assumed to be able to accurately estimate the locations of their users, while the latter is assumed to know the infection status of a country's citizens. Through exchanging these data, contacts with positive individuals can be detected. As both location data and infection status are highly sensitive, they cannot be exchanged in the clear. To ensure privacy, we propose a system allowing MOs and GA to perform computations directly on encrypted data. The underlying cryptographic scheme of our solution is based on fully-homomorphic encryption techniques that allow computations on encrypted data without knowledge of the decryption key, allowing outsourcing computations even to potentially untrusted parties. Our protocol uses the Cheon-Kim-Kim-Song (CKKS) cryptographic system to compute a score of infection likelihood, given the number of positive individuals in proximity within a given period. As computation of this score in encrypted domain is highly time-consuming, we propose approximations of several key operations performed. In doing so, reductions in computational time are achieved, at the expense of error in values of the score. Through simulations, the error versus computational time trade-off is evaluated. Results show computational times of up to several tens of minutes on a personal computer, with worst-case induced absolute errors of 0.6634.

Keywords: Privacy; Homomorphic Encryption; Network-Based Positioning; Covid19

Abstract in lingua italiana

L'epidemia di Coronavirus dalla fine del 2019 all'inizio del 2020 ha portato una serie di sfide per l'umanità. Il tracciamento dei contatti consente di diminuire l'ulteriore diffusione del contagio. Tradizionalmente, i sistemi di tracciamento si basano su applicazioni telefoniche che verificano la vicinanza con individui positivi al virus e avvisano gli utenti di potenziali rischi. L'adozione di soluzioni basate su app è stata scarsa, limitandone l'efficacia. Le ragioni principali sono la poca esperienza tecnologica di larga parte della popolazione e la poca disponibilità a installare le app sui propri telefoni. Inoltre, le app sono state percepite come un rischio per la privacy, ostacolandone ulteriormente l'adozione. Per superare questi problemi, proponiamo un sistema di tracciamento dei contatti basato sulle tecnologie di localizzazione delle infrastrutture telefoniche (e.g., 5G), che garantiranno un errore sotto al metro nei prossimi anni. Il sistema proposto prevede la cooperazione tra Operatori di telefonia mobile (MO) e Agenzie governative (GA). Si assume che il primo sia in grado di stimare con precisione la posizione dei propri utenti, e il secondo conosca lo stato di infezione dei cittadini di un paese. Attraverso lo scambio di questi dati si possono rilevare i contatti con individui positivi. Poiché sia i dati sulla posizione che lo stato dell'infezione sono altamente sensibili, non possono essere scambiati in chiaro. Per garantire la privacy, proponiamo un sistema che consenta a MO e GA di eseguire calcoli direttamente su dati crittografati. Lo schema crittografico alla base della nostra soluzione, di tipo fully homomorphich, permette calcoli su dati crittografati senza la conoscenza della chiave di decifratura, consentendo l'esternalizzazione dei calcoli anche a parti potenzialmente non attendibili. Il nostro protocollo utilizza il sistema crittografico Cheon-Kim-Kim-Song (CKKS) per calcolare un rischio di infezione, dato il numero di individui positivi nelle vicinanze in un determinato periodo. Data l'elevata complessità del calcolo nel dominio cifrato, proponiamo approssimazioni di diverse operazioni chiave eseguite. In tal modo, si ottengono riduzioni del tempo di calcolo, a scapito dell'errore nei valori del rischio. Attraverso simulazioni, viene valutato il trade-off tra errore e tempo computazionale. I risultati mostrano tempi di calcolo fino a diverse decine di minuti su un personal computer, con errori assoluti indotti nel caso peggiore di 0,6634. **Parole chiave:** Riservatezza; Crittografia omomorfica; Posizionamento basato sulla rete; Covid19

Contents

1 | Introduction

The initial scare and massive spread of the COVID-19/SARS-CoV-2 coronavirus and its related disease in late 2019–early 2020 put a significant strain on everyone all over the world. Starting from the first half of 2020 and essentially until working versions of the vaccines were successfully deployed, tracing the contacts of the infected individuals was the main way of protecting the general public, save for the highly restrictive curfews and lockdown mandates.

The present work is inspired by a network-based contact tracing protocol described in [18] and [19], which use two complementary cryptographic systems, namely the Pailler partially-homomorphic encryption scheme [67] and Shamir secret sharing [75] for communications between the involved parties — namely, the MOs, their Users, and the GA. These two schemes used in parallel produce a relatively high communication overhead per User. This thesis proposes an alternative network-based approach using a fully-homomorphic encryption (FHE) system to reduce said communication overhead and potentially allow computations to be performed in a highly distributed fashion without the need for trusted elements.

1.1. Overview and Motivation

Contact tracing in the status quo of the Covid-19 pandemic has been done via various mobile applications. These applications use either a decentralized approach based on the proximity of users, or a centralized approach based on evaluation of user locations (visualization in figure 1.1, as shown in [51]). The proximity approach evaluates how close users are to one another to assess the risk of infection based on privately-exchanged tokens (generally using Bluetooth). On the other hand, the location-based approaches track user locations and compare them to the locations of known infected users. Both location-based and token/Bluetooth-based solutions have potentially-unresolved vulnerabilities.

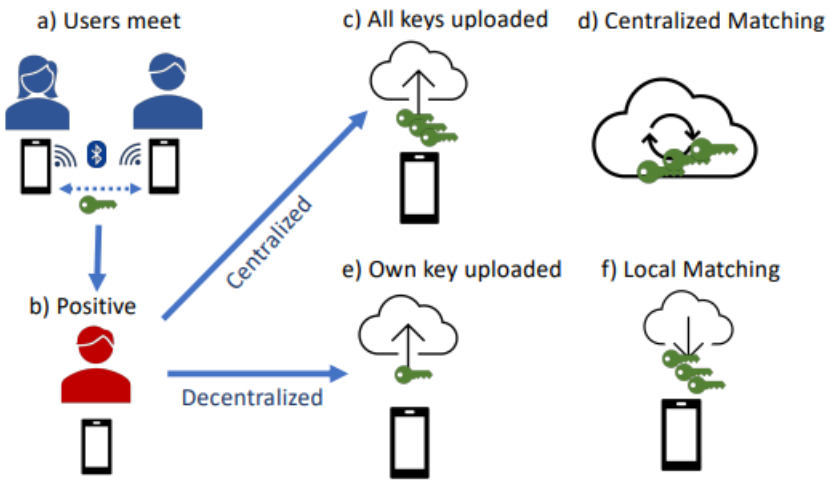


Figure 1.1: Centralized versus decentralized contact tracing [51].

Location-based solutions (e.g. Saudi Arabia's Tawakkalna, Ghana's GH COVID-19 Tracker, Israel's HaMagen 2.0 etc.) are susceptible to re-identification attacks ([35, 63]), in which the identities of infected individuals can be obtained from their shared locations. In comparison, token-based applications (e.g. Italy's Immuni, France's TousAntiCovid, Australia's COVIDSafe, Singapore's TraceTogether etc.) might allow a number of crafted queries and token manipulation ([35]) to obtain infection status for a specific person based on contact time.

The battery consumption related to leaving the GPS and/or Bluetooth functionalities running at all times on users' phones might also deter a number of users from participating to the contact tracing effort via applications (especially for GPS-based solutions).

According to [52], a critical mass of $\sim 80\%$ of all smartphone users or $\sim 56\%$ of the whole population has to be part of the same contact tracing scheme to ensure that the epidemic can be suppressed. According to Google-collected data on contact tracing applications ([44], [45], [46]). By 25.01.2021 [45], roughly a year after the start of the pandemic, only 2 countries had reportedly reached a desirable penetration (Singapore's TraceTogether reaching 80%, and Qatar's Ehteraz reaching 91%), while in the US only D.C.'s DC CAN managed to barely reach a 56.1% penetration. A similar report from 10.06.2020 [44] shows that no country or US state managed to reach over 18% of the target population. The reasons for this lack of adoption are either a distrust for the application-based solutions or a lack of sufficient technological knowledge. The distrust might stem from any of the above-mentioned reasons — either not wanting the battery to be drained or fear of potential exposure of private data, such as location or health information.

Given that mobile operators already estimate user locations to improve service quality

[17], and that 5G location can be accurate within 1 metre in dense and ultra-dense networks [57], a network-based solution for contact tracing seems, if not the most logical step, at least one to be considered. Such a solution can be implemented without any need for direct user interaction, hence eliminating the need for any tech-savvy skills from the side of the users. The added battery consumption caused by keeping various services on at all times will also be eliminated. The only remaining concern is security. Refs. [18, 19] propose a solution in which mobile operators (MO) and a government agency (GA) collaborate to share data privately using a combination of Pailler [67] partially-homomorphic encryption and Shamir [75] secret sharing. This work attempts to improve the underlying protocol by using only one fully-homomorphic encryption (FHE) system. FHE is a long-theorized concept in cryptography [72] made possible in 2009 by Gentry's seminal doctorate thesis [41]. FHE is an encryption scheme that enables analytical functions to be run directly on encrypted data while yielding the same encrypted results as if the functions were run on plaintext. The desired improvements are added security (even against attacks via quantum algorithms), reduced communication overhead, as well as eliminating the need for a trusted party. The reduced communication overhead is a result of data being only exchanged once.
In this thesis, we propose a prototype protocol of such a network-based contact tracing scheme using FHE and analyze its performance.

1.2. Thesis Structure

This thesis is structured into 5 Chapters, Introduction notwithstanding. Chapter 2 presents related work in the fields of privacy-preserving technologies and location accuracy-enabling scientific developments. Chapter 3 presents the theoretical results used in support of the protocol construction, both in terms of encryption and in terms of geo-localization methods. Chapter 4 describes in detail the protocol, its equivalent without encryption, the approximations made and their respective justifications. Chapter 5 presents the simulations performed and interprets the results. Chapter 6 presents conclusions and ideas for future developments. A more detailed description of each Chapter is presented in the following.

Chapter 2 presents an overview of related work, both in terms of contact tracing technologies, as well as privacy-preserving technologies that facilitate the protocol and allow a certain number of assumptions to be made. The contact-tracing Section 2.1 is split into subsections dedicated to proximity-based (subsection 2.2) and location-based (2.2.1) implementations, respectively. Both provide a number of examples from the two approaches,

along with points of divergence with the current work. The second Section 2.3, dedicated to privacy-enhancing technologies, is dedicated to listing the advances in FHE, both as theoretical schemes, as well as practical applications, that facilitate the scope of this thesis.

Chapter 3 explains the theory behind the proposed system, with a first Section describing the used cryptographic system and a second Section dedicated to positioning technologies. Within Section 3.1, the theoretical background of FHE is presented, starting from initial theories, evolution of schemes, and ending with a detailed description of the Cheon-Kim-Kim-Song scheme (CKKS) [26]. Within Section 3.2, the case is made for the potential to perform accurate location estimation in dense and ultra-dense 5G or 6G mobile networks, as a means of collecting user locations with sub-metre accuracy. It presents promising combinations of algorithms and input data in a 5G setting that would enable the locations used in the protocol computations to be relevant for the real-life situations considered.

Chapter 4 describes the proposed protocol in more detail, with Section 4.1 dedicated to its architecture — involved entities, their roles and the proposed attacker model — Section 4.2 dedicated to the description of the protocol functionality without encryption, and final Section 4.3 describing the implementation in cipher domain. Protocol architecture Section 4.1 describes in detail the entities involved in the protocol in terms of their knowledge and roles, establishes privacy goals, and the attacker model considered. The equivalent functionality Section 4.2 presents the contact and score evaluation procedures as they would occur without the need for preserving privacy. Section 4.3 concerning implementation in cipher domain contains a subsection on the considered cryptographic primitives, one on the necessary adaptations from the plaintext version, and a final one on the final protocol.

Chapter 5 is dedicated to simulation and results. It starts by establishing a Gauss-Markov mobility model for generating User locations in Section 5.1. Then, the considered performance metrics are described in Section 5.2. Section 5.3 presents the setup of the simulation process. Finally, Section 5.4 presents and interprets the produced results.

Chapter 6 represents the conclusion and future work. In conclusion, the main ideas of the thesis are reiterated. For the future work Section, a number of directions for future research and developments are identified.

2 | Related Work

In this chapter, we present the literature related to the main subjects of the thesis, namely contact tracing and fully-homomorphic encryption. Section 2.1 provides an overview of used contact tracing solutions, with In section 2.3, the FHE state of art landscape is presented. It provides an overview of current FHE schemes and implementations, while also showing a list of cutting-edge applications and facilitating technologies. At its end, a short nod is given to the specific programming libraries used in this work.

2.1. Contact Tracing Solutions

This section presents a short overview of contact tracing approaches. Traditionally, contact tracing is performed through mobile applications. These applications use the of the users and confront them with the known infected user data, generally either through collecting GPS data and comparing it with the (assumed) known locations of infected individuals, through leveraging Bluetooth interactions between phones and evaluating proximity with known infected users, or a combination of the two. Subsection 2.2 provides details on the proximity-based mobile applications used to perform contact tracing, while 2.2.1 serves as a glimpse into location-based approaches.

2.2. Proximity-Based Solutions

Proximity-based solutions detect whether users are close to one another and perform contact tracing by keeping private tokens for all individuals and storing the tokens of all contacts. Then, when an individual becomes infected, it is flagged in a database and during an exchange with said database, all other users who were in contact with the infected individual are notified.
Of the proximity-based contact tracing solutions, we mention Singapore's TraceTogether, Mexico's CovidRadar, Philippines' StaySafe and Italy's Immuni. Outstandingly, while not being generically tagged as using Bluetooth or GPS, some apps are DP3T-compliant [79] — a joint effort of a number of European experts (and Dan Boneh) to produce a

privacy-preserving contact tracing framework — while some apps use the Google-Apple exposure notification service [47, 48], which is essentially also Bluetooth-based. Google's Covid-19_apps [44], Covid Tracing Tracker [45], and COVID-19 Digital Rights Tracker Supporting Data [46] spreadsheets were helpful to get a bird's eye view of the state of the app-based tracing at different points in time — the former two stopped being updated at a cutoff point and essentially serve as snapshots.
The main drawbacks of proximity-based solutions are the vulnerabilities to attacks identified in [35], as well as the need to keep the Bluetooth on at all times, which comes at an energy cost.

2.2.1. Location-Based Solutions

Location-based solutions use GPS to collect user location data and compare them against the locations of known infected users. Of the application-based location solutions, we mention China's health code system, Bulgaria's Virusafe, Kuwait's Shlonik, Cyprus' CovTracer and Iceland's Rakning C-19.
Network-based approaches we will consider location-based, as contact tracing is performed after assessment. To the best of our knowledge, except for [18, 19], there was only one other similar work that discussed contact tracing through WiFi sensing in enterprise and campus networks [78]; said work additionally proposes an algorithm that scales to networks of tens of thousands. For the purposes of this thesis, the drawback of this approach is the fact that it considers authenticated networks with multiple access points. This would not be translatable to outdoor movements, where WiFi connections are scarce or nonexistent.
The two works [18, 19] essentially paved the way for this work, by proposing a mobile operator-based network approach and a general protocol architecture which has inspired the current architecture — the participating entities are modeled in a similar way and the risk evaluation is also done in similar fashion, while the specific cryptographic system used is a combination of more traditional cryptographic approaches, such as Pailler's probabilistic system [67] and Shamir's secret sharing scheme [75]. Notably, some entities are also given more trust in the security sense of the concept — i.e., if they break, the system becomes insecure.
The protocol proposed in this thesis essentially falls into the category of location-based contact tracing solutions, as proximity between users is only evaluated after having gathered their locations.

2.3. Privacy-Enhancing Methods

Fully homomorphic encryption methods are developed with secure data sharing in mind. This is particularly relevant for the subject of this work, as it involves sensitive data sharing also. A number of frameworks and platforms are currently considered state-of-the-art. Of the frameworks, we recall some alternative implementations of the Cheon-Kim-Kim-Song cryptosystem. These include:

- HElib [50]: Developed by IBM, it uses `C++` to implement the Brakerski-Gentry-Vaikuntanathan (BGV) [24], as well as the CKKS [26] schemes;
- Microsoft SEAL [74]: Developed by Microsoft, it uses `C++` to implement the Brakerski-Fan-Vercauteren [39] (BFV) scheme, as well as BGV and CKKS;
- HEAAN [12, 26]: Short for "Homomorphic Encryption for the Arithmetic of Approximate Numbers"; developed by the cryptography group at Seoul National University as a proof of concept for the CKKS cryptographic system;
- Lattigo [13, 66]: Developed in Go by École Polytechnique Fédérale de Lausanne's Laboratory for Data Security (EPFL-LDS), it implements the BFV, BGV and CKKS schemes, with a focus on multi-party computation;
- PALISADE [7, 14]: Developed collaboratively by a number of universities and companies, it uses `C++` to provide a general-purpose library for most fully-homomorphic intents and purposes. It implements BGV, BFV, CKKS, FHEW, TFHE and multi-party computation extensions for BFV, BGV and CKKS. It got absorbed into the OpenFHE project [20] mid-2022, along with select features of HElib and HEAAN.

In terms of platforms, the most relevant for this work are the ones which provide secure sensitive data protection (such as location or health information), or works which might improve the performance of

- Inpher's XOR platform [5]: Bleeding-edge secure multi-party computation enterprise platform, it also enables secure federated learning. Inpher claims XOR will implement FHE when hardware acceleration becomes available. It is currently used inside Amazon Web Services and Google Cloud;
- Zama's Concrete framework [9, 15]: Open-source Rust implementation of the TFHE cryptographic system, constantly improved. Apart from the base library, it has three additional flavours: a machine learning-oriented one implemented in Python, one designed with hardware acceleration in mind, and finally another Python implementation with numpy-like capabilities. Concrete-Numpy was briefly considered as

a coding base for simulation in this work, but ultimately abandoned due to the limitations regarding size of numbers used in calculations (i.e., at most 7 bits with limited accuracy at the time);

- Duality's SecurePlus framework [3]: Enterprise solution for collaborative privacy-preserving projects. It uses homomorphic encryption for analytics, machine learning and SQL-like query functionalities. It supports integration with OpenFHE;
- Enveil's ZeroReveal framework [4]: Enterprise solution with two submodules, one for encrypted search and one for machine learning. The encrypted search submodule boasts military-grade encryption;
- Google's FHE Transpiler [49]: Modular platform that allows conversion from code working on plain data to code working on encrypted data. It translates high-level language code into another high-level language code that performs the same operations under a FHE library API. It boasts interchangeability of both initial and final languages, as well as of the used FHE system.
- Cornami's TruStream [2]: Software-defined computer architecture that boasts $10^6\times$ better performance in FHE situations.

In terms of used cryptography libraries, Saroja Erabelli's py-fhe Python library [8] implements CKKS and BFV efficiently and has been used as the coding base for this work. The main motivation for this is the scarcity of Python implementations for FHE, a programming language better-suited for logging execution times of code fragments. The second motivation is the freedom and clarity for setting cryptographic parameters within the library.
Another library used for the initial code in this work was PySEAL [10]. The library is an implementation of Microsoft's SEAL v2.3 in Python. Unfortunately, at the time, SEAL only offered support for BFV, which quickly turned out to be a cryptographic system unsuitable for comparing numbers larger than 64.
To the best of our knowledge, this is the first attempt at implementing FHE in contact tracing applications. From the point of view of contact tracing, this work expands upon some previous explorations into network operator-based contact tracing [18, 19], while giving the government authorities less responsibilities and hence less trust. From the point of view of applying cryptography, this work is fresh in the sense that it provides an insight into the feasibility of FHE, particularly regarding contact tracing. Moreover, it uses an approximation of a step function via additions and multiplications to circumvent the lack of comparison capabilities within the currently available FHE schemes.

3 | Theoretical Background

In this chapter, we present the theoretical background required to understand the proposed solution. Within Section 3.1, a brief history and theoretical background of (FHE) is given. In subsection 3.1.1, a number of different FHE schemes are mentioned, along with the most relevant advances and features. In subsection 3.1.2, special attention is given to the Cheon-Kim-Kim-Song cryptographic system [26], which made the basis for this work. Within the final Section 3.2, a number of algorithms and technologies that allow sub-metre positioning in 5G networks are reviewed. Short descriptions of input data and algorithms are given to both conventional approaches, as well as machine-learning implementations, with a mention of the projected improvements in the upcoming 6G wireless era.

3.1. Homomorphic Encryption Descriptions

Encryption schemes allow data encryption in such a way that only the decryption key owner can access data in plaintext. Schemes with homomorphic properties — i.e., homomorphic encryption (HE) schemes — allow performing a set of operations directly on encrypted data, without the need to decrypt and re-encrypt. More formally, HE is a way of encrypting data such that the encryption function defines a group homomorphism between the plaintext space and the ciphertext space. That is to say, having $(P, *)$ the group of plaintexts and $(C, \circ)$ the group of ciphertexts with their respective operations $*$ and $\circ$, the encryption function $Encr : P \mapsto C$ behaves such that $Encr(p_1 * p_2) = Encr(p_1) \circ Encr(p_2), \forall\ p_1, p_2 \in P$.

Two main types of HE schemes exist — namely the partially-homomorphic encryption schemes, and the FHE. The former allows performing only a set of operations on ciphertexts, while the latter allows all operations to be performed. Of the former, examples are the RSA cryptographic system [71] and the Pailler cryptographic system [67]. In the default version of RSA, multiplications between ciphertexts yield another ciphertext, which encrypts the product of the initial plaintexts (homomorphism under multiplication). In the Pailler scheme, a multiplication between ciphertexts encrypts the sum of the initial

plaintexts (homomorphism between $(P, +)$ and $(C, \cdot)$).
FHE was first formally proposed in Ref. [72]. In this work, Shamir, Adleman and Dertouzos theorised a FHE system. For this, they considered two algebraic systems

$$\begin{aligned} S &= \langle A; f_1, \ldots, f_n; p_1, \ldots, p_m; c_1, \ldots, c_t \rangle \\ S' &= \langle B; f'_1, \ldots, f'_n; p'_1, \ldots, p'_m; c'_1, \ldots, c'_t \rangle \end{aligned} \tag{3.1}$$

Where:

- A, B are sets of values — e.g., integers $\mathbb{Z}$;
- $f_1, \ldots, f_n$ are operations — i.e., when applied to one or more values in A, they yield a value in A, $f_i(a_1, \ldots, a_k) = a_x$ (such as $+, -, \cdot, \div$ for $\mathbb{Z}$). So are $f'_1, \ldots, f'_n$ for B;
- $p_1, \ldots, p_m$ are predicates — i.e., when applied to one or more values in A, they yield a truth value, $p_i(a_1, \ldots, a_j) = 0/1$ (such as $\leq$). So are $p'_1, \ldots, p'_m$ for B;
- $c_1, \ldots, c_t$ are distinguished constants in A (such as 0, 1 for $\mathbb{Z}$). So are $c'_1, \ldots, c'_t$ for B.

These algebraic systems are mapped onto one another algebraic system via an invertible function $\phi : S \mapsto S'$. This function, along S and S', denote a FHE scheme if operations, predicates and distinguished constant in S are carried by ϕ into S' in such a way that the results are preserved after decryption. Formally:

$$f'_i(\phi(v_{a_1}), \ldots, \phi(v_{a_k})) = \phi(f_i(v_{a_1}, \ldots, v_{a_k})), \forall i \in \{1, \ldots, n\}, v_{a_1}, \ldots v_{a_k} \in A; \tag{3.2}$$

$$p'_i(\phi(v_{b_1}), \ldots, \phi(v_{b_j})) = \phi(p_i(v_{b_1}, \ldots, v_{b_j})), \forall i \in \{1, \ldots, n\}, v_{b_1}, \ldots, v_{b_j} \in A; \tag{3.3}$$

$$\phi(c_l) = c'_l, \forall l \in \{1, \ldots, t\}. \tag{3.4}$$

In addition, the scheme would have to be secure against ciphertext-only attacks, chosen plaintext attacks, and the operations and predicates in encrypted domain should not be enough to yield an efficient computation of ϕ.

3.1.1. Fully-Homomorphic Encryption Schemes

The idea of a fully-homomorphic cryptographic system remained theoretical for 31 years after the paper was published, until Craig Gentry proposed a feasible scheme in his PhD thesis [41]. Within it he proposed a means of encrypting numbers with added noise, and the scheme's security was based on Regev's lattice and learning with errors (LWE) [70] results. More specifically, the LWE problem is a search problem that requires solving a set of linear equations, each correct within an additive error term. In his work, Regev proved the problem to be equivalent to several worst-case hard (either proved or conjectured) lattice problems. Differently from most of the widely-used cryptosystems (e.g., RSA), FHE systems based on the LWE problem have been proven secure even against attacks performed using quantum computing technologies.

In Gentry's seminal thesis, he proposed a scheme that became the origin of noise-based FHE systems. In such systems, ciphertexts are each associated with some level of noise. At each operation involving the respective ciphertext, the noise amount increases. The ciphertext remains decryptable until the noise reached a threshold or noise budget over which the initial plaintext is unrecoverable. While additions and subtractions are almost "free" in terms of noise increase, multiplications increase the noise significantly. Hence, there is a limited number of multiplications one can perform on ciphertexts before hitting this noise budget. The main result that turns this contruction into a FHE scheme is the bootstrapping procedure, through which ciphertext noise is reset and the initial plaintext is encrypted as a fresh ciphertext under a new secret key. This enables the multiplications to continue essentially ad infinitum. However, it is computationally very expensive and time-consuming.

In 2009, this marked the beginning of what is generally considered the first generation of FHE schemes, with a first implementation in 2009 [37] and a second one [42] in 2010/11 taking over 30 minutes for performing basic bit operations.

The second generation of fully-homomorphic encryption schemes was based on either the Ring Learning with Errors problem (RLWE) — which is a subcase of LWE specialized to polynomial rings over finite fields — or on the NTRU problem [53], whose security has since been proven flawed. The most notable representatives of the second generation are the BFV [39] and the BGV [24] cryptographic systems, both working on integers and still being considered state of the art today due to precision and security considerations. BGV was based on the work of Zvika Brakerski and Vinod Vaikuntanathan [23], while BFV was developed with Brakerski's work [22] in view. Notably, they both provide single instruction multiple data (SIMD) functionalities, albeit for shorter vector lengths than more recent generations. While bootstrapping is possible in these schemes, emphasis was

put on calculations being done without performing it and hence them acting as levelled homomorphic schemes — i.e., the ciphertexts are considered as having a limited number of subsequent multiplications or max level, and after every one their level increases.
After the second generation of FHE, the advances have branched out into two distinct directions. One of the directions was towards faster bootstrapping and generally quicker computations, even at the cost of reduced ciphertext sizes, while the other was focusing on implementing SIMD schemes that support complex numbers and floating point operations. Although not necessarily different generations in terms of time, the former are considered to be in the third generation, while the latter currently only has the Cheon-Kim-Kim-Song (CKKS) [26] scheme as representative.
The third generation was kickstarted by the GSW scheme [43], which proposed a new relinearization procedure for dealing with multiplications in cipher domain, as well as a new evaluation scheme which did not require the previously-used evaluation key (i.e., a chain of the user's encrypted secret keys) for each user. The relinearization of ciphertexts is a process during which 3-dimensional ciphertexts resulting from homomorphic multiplications are turned back into 2-dimensional ciphertexts — by means of a relinearization key — on which further operations can be performed. The scheme was then improved by Léo Ducas and Daniele Micciancio to create the FHEW scheme [38], which reduced bootstrapping times to under a second. A further improvement came in the shape of TFHE [32], which employed a novel underlying field to further decrease bootstrapping times to under 0.1 seconds and reduce bootstrapping keys from 1 GB to 16 MB.
The CKKS scheme is described in detail in the following subsection.

3.1.2. CKKS

The first and as of yet the only representative of the fourth generation of fully-homomorphic encryption schemes, the CKKS scheme [26] is the only scheme that supports computation over complex numbers, while also providing support for SIMD functionalities over encrypted values and providing an efficient rescaling procedure for securely reducing error and ciphertext size. As this scheme was chosen for usage as a theoretical base in this work, a comprehensive description of its functionality will be given.
CKKS is a leveled homomorphic encryption scheme, in the sense that certain operations in the encrypted domain will only be possible between ciphertexts on the same "level". Freshly encrypted ciphertexts start on level 0 and increment in level as they are subsequently multiplied with ciphertexts of the same level (i.e., two ciphertexts of level n multiplied homomorphically will produce a ciphertext of level $n+1$). Moreover, the noise budget is limited, and multiplications increase the noise significantly — roughly, the noise

size in bits after the multiplication is equal to the sum of the noise sizes in the factors. Through bootstrapping [28], a potentially infinite number of subsequent multiplications can be performed, and the scheme becomes fully-homomorphic. Predictably, this is time-consuming and while using CKKS as the underlying scheme of the protocol we prefer to avoid it and alternatively set parameters to stay within the maximum level.
In the following lines, the differentiating aspects of the scheme are described. Plaintext information is kept in the most significant bits (MSB) of the ciphertext with error in the least significant bit (LSB) portion, and as subsequent multiplications are performed, the MSB portion is pushed towards the end of the available ciphertext size, with error occupying the rest (see 3.1). One of the novelties of the CKKS scheme is the rescaling operation. It is performed by dividing both the ciphertext and its modulus by the same division factor — usually equal to the scaling factor used in encoding and rounding to the nearest valid ciphertext. This cuts both the final bits of the ciphertext, as well as the final bits of the error, which makes both the error blowup during multiplications more manageable and less likely to cause major inaccuracies due to error during decryption, as well as make any follow-up operations on the ciphertext more efficient due to smaller size.

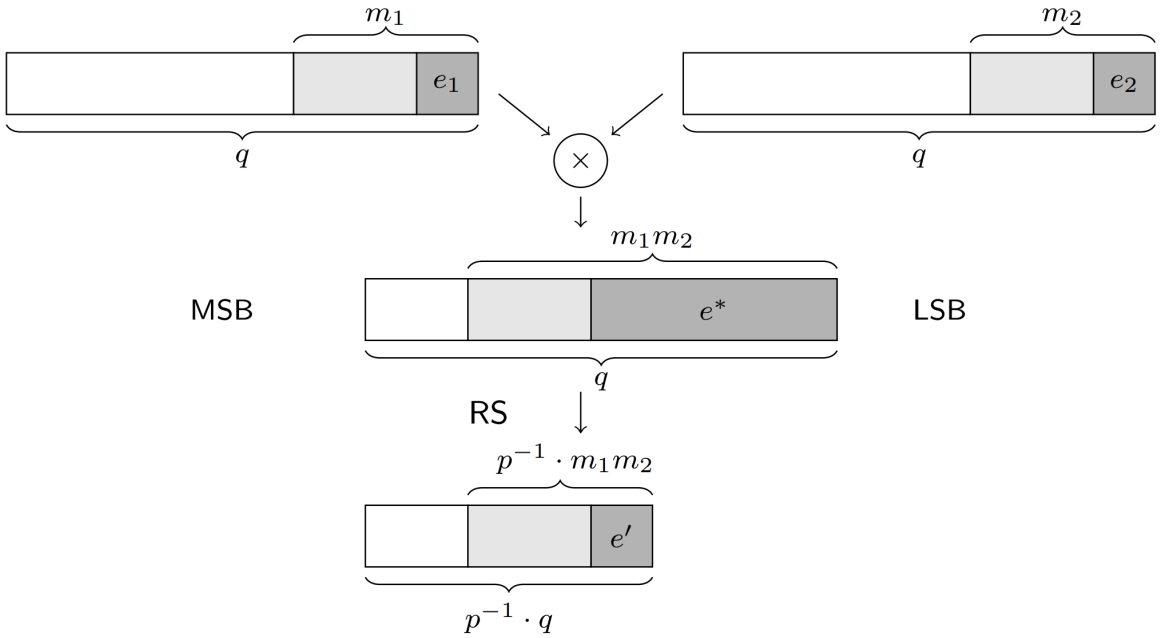


Figure 3.1: Multiplication and rescaling in CKKS [26].

In the following subsection, the operations in the scheme will be described. The ciphertext lifecycle follows a patter of encode–encrypt–perform operations–decrypt–decode. Encoding is needed prior to encryption because the initial plaintexts are vectors of complex numbers, and the ciphertext space is a polynomial field of high degree with coefficients

bounded by a very large number. Some parallels with practical implementation aspects will be drawn, mainly by exemplifying with respect to the library used in simulations — Saroja Erabelli's (formerly of Massachusetts Institute of Technology, now of Duality Technologies Inc.) py-fhe Python implementation of the scheme.

Plaintext Encoding

Encoding maps the plaintext represented by a vector of 2^{k-1} complex numbers with integer real and imaginary parts onto a polynomial ring $\mathbb{Z}/(X^{2^k}+1)$ with coefficients bounded by some 2^n. The full mathematical description, while interesting, is beyond the scope of this thesis.
The main relevant idea is that encoding is extremely quick compared to the speed of multiplications, and plaintext vector size determines ciphertext polynomial degree. Moreover, before embedding the plaintext into the ciphertext polynomial ring, all values in the plaintext vector are multiplied by a scaling factor Δ.

Encryption

The encryption routine suite has three major steps: key generation, encryption and decryption. The fixed parameters prior to generating the encryption context are:

- quotient polynomial degree N,
- scaling factor Δ;
- fresh ciphertext modulus q;
- big modulus P (in practice, of size $\frac{5}{4}q$ or bigger).

The latter is used for performing modular reduction during the relinearization process.

- Key generation: secret key sk sampled according to parameters N (ciphertext polynomial degree, and also plaintext vector length) and h (hamming weight). The resulting secret key is:

$$sk \leftarrow HWT(h) \tag{3.5}$$

  where $HWT(h)$ is the set of signed binary vectors $\{0, \pm 1\}^N$ with Hamming weight h. In practice, the chosen value for h is $h = N/4$, to keep a sparse secret key so as not to have computational complexity blow up during encryption. The produced key is a vector of the same length as the polynomial degree/ plaintext vector length, with a quarter of the values ± 1;

The public key pk is produced according to:

$$pk \coloneqq (b, a) \in \boldsymbol{R}_q^2 \tag{3.6}$$

where $a \leftarrow \boldsymbol{R}_q$ is uniformly sampled from $\boldsymbol{R}$ with coefficients bounded by q, $e \leftarrow G(\sigma^2)$ is an error term sampled from a discrete Gaussian distribution with standard deviation σ, and $b \coloneqq -as + e \pmod q$. The public key is a pair of polynomials — to be read as *vectors of length* N — of which one is a fully random term, and the other is a linear combination of the secret key and error;
The relinearization key rk (called evaluation key in [26]) is produced by:

$$rk \coloneqq (b', a') \in \boldsymbol{R}_{Pq}^2 \tag{3.7}$$

where P is a big integer, $a' \leftarrow \boldsymbol{R}_{Pq}$ sampled uniformly from $\boldsymbol{R}_{Pq}$, $e' \leftarrow G(\sigma^2)$ error sampled from a Gaussian distribution with standard deviation σ, and $b' \coloneqq -a's + e' + P \cdot sk^2$. For the purposes of the protocol, it is only relevant that the relinearization key is very big.

- Encryption:

$$\mathrm{Enc}_{pk}(m) \coloneqq v \cdot pk + (m + e_0, e_1) \pmod q \tag{3.8}$$

where $v \leftarrow \mathcal{ZO}(0.5)$ ($\mathcal{ZO}(\rho)$ samples each entry in the vector from $\{0, \pm 1\}^N$, with probability ρ for both ± 1 and $1 - \rho$ for 0), $e_0, e_1 \leftarrow G(\sigma^2)$ sampled from a discrete Gaussian distribution with standard deviation σ;

- Decryption:

$$\mathrm{Dec}_{sk}(\boldsymbol{c} = (b, a)) \coloneqq b + a \cdot sk \pmod{q_l} \tag{3.9}$$

where q_l is the modulus of the ciphertext level. Of note is the linearity of the decryption procedure.

Operations on Ciphertexts

The considered operations on ciphertexts are addition with ciphertext and plaintext, multiplication with ciphertext and plaintext, and rescaling. Of these, most interest is presented by the multiplications — as is the case in any FHE scheme — as well as the rescaling. As a note before going forward, the plain values taken in the following computations are actually encodings of plaintext vectors.

- Plain addition:

$$\text{Add\_plain}(\boldsymbol{c} \coloneqq (b, a), \boldsymbol{p}) = (b + \boldsymbol{p}, a) \pmod{q_l} \tag{3.10}$$

where $\boldsymbol{c}$ is a ciphertext of level q_l and $\boldsymbol{p}$ is an encoding of a vector of plain values. The scaling factor of the two operands must match in order to ensure that the plain value is not added to the error part of the ciphertext. Given that the plaintext is a fresh encoding, that means the scaling factor of the ciphertext term must be Δ;

- Addition:

$$\text{Add}(\boldsymbol{c_1}, \boldsymbol{c_2}) \coloneqq \boldsymbol{c_1} + \boldsymbol{c_2} \pmod{q_l} \tag{3.11}$$

where $\boldsymbol{c_1}, \boldsymbol{c_2}$ are both ciphertexts of level q_l. The moduli and the scaling factors of the ciphertext terms have to match;

- Multiplication with plain:

$$\text{Mult\_plain}(\boldsymbol{c} = (c_0, c_1), \boldsymbol{p}) = (c_0 \cdot \boldsymbol{p}, c_1 \cdot \boldsymbol{p}) \pmod{q_l}. \tag{3.12}$$

Of note is that the plain value is scaled up by Δ and hence the result will be scaled by an additional Δ;

- Multiplication:

$$\begin{aligned} \text{Mult}_{rk}(\boldsymbol{c_1} = (b_1, a_1), \boldsymbol{c_2} = (b_2, a_2)) &\coloneqq (d_0, d_1) + \lfloor P^{-1} \cdot d_2 \cdot rk \rceil \pmod{q_l} \\ (d_0, d_1, d_2) &= (b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2) \pmod{q_l} \end{aligned} \tag{3.13}$$

where $\lfloor \cdot \rceil$ represents the rounding operation. In similar fashion to the plaintext multiplication procedure, the resulting ciphertext will be scaled by the product of the scaling factors of its factors. Moreover, the moduli of the initial ciphertexts have to match;

- Rescaling:

$$\text{RS}_{l \to l'}(\boldsymbol{c}) \coloneqq \lfloor \frac{q_{l'}}{q_l} \boldsymbol{c} \rceil (\text{mod } q_{l'}) \tag{3.14}$$

where l is the level of of the initial ciphertext $\boldsymbol{c}$, l' is the level of the final ciphertext, and $q_l, q_{l'}$ are the respective ciphertext moduli. In practice, rescaling is done by providing a division factor rather than a level.

Parameter Selection

In practice, the cyclotomic polynomials chosen as parameters are of form X^{2^k} for some $k \in \mathbb{N}$, for both ease of computation as well as ensuring maximal size of plaintexts and ciphertexts — this blends security with practicality. The resulting vector will then be of size $poly\_deg = 2^{k-1}$, and have the real parts encoded within the first half of the polynomial, and the imaginary parts encoded in the second. Practical implementations then select a sparse secret key sk — so as not to make encryption too computationally intensive — of Hamming weight $h = poly\_deg/4$.

Security-wise, the de facto standard for all fully-homomorphic encryption schemes is ensuring the system is computationally resilient against chosen plaintext attacks (CPA) — for CKKS, also considering the CPA+ variant [59], in case the decrypted data are then shared with potentially malicious parties — as well as a number of lattice-based attacks [31].

In this work, the considered desired security level is $\lambda = 128$ bits. That is, to be able to break the encryption, an attacker would have to perform 2^{128} operations. The first parameter to be chosen, following the security level, is $poly\_deg$. This implicitly will set sk Hamming weight to $h := \frac{poly\_deg}{4}$. For secret key hard to guess under security level λ, given the key generation formula 3.5, $h \geq 64$, and hence $poly\_deg = 256$. Next, ciphertext modulus q, strongly coupled with initial plaintext maximum size $\log N$ are chosen in accordance to desired security level. Finally, scaling factor Δ is chosen in accordance to desired maximum level of ciphertexts (i.e., maximum required number of multiplications) and final result precision. The guidelines for choosing parameters in accordance to security and desired functionalities are given as a series of tables in [31] (table 3.1).

CKKS Parameter Selection for Security against Lattice-based Attacks

h	log$p$	log$N$	logPq	logΔ	max. level
64	10	14	219	29	5
		15	431	33	9
		16	930	42	17
		17	2022	54	29
	20	15	431	41	7
		16	930	49	14
		17	2022	60	25
	30	16	930	57	12
		17	2022	67	22
		15	431	33	9
128	10	14	337	31	7
		15	700	38	13
		16	1450	48	23
		17	2900	62	36
	20	14	337	39	5
		15	700	46	11
		16	1450	55	20
		17	2900	68	32
	30	15	700	54	9
		16	1450	62	17
		17	2900	75	29
256	10	13	195	28	4
		14	393	33	8
		15	821	40	15
		16	1623	50	24
		17	3300	65	39
	20	14	393	41	6
		15	821	47	12
		16	1623	57	21
		17	3300	57	21
	30	14	393	50	5
		16	1623	65	19
		17	3300	78	32

Table 3.1: CKKS Parameter Selection.

Table 3.1 shows a table of the practical parameter selection for security against IND-CPA+ and lattice attacks, as shown in [31]. Assuming security parameter $\lambda = 128$ bit: h — Hamming weight of secret key, $\log p$ — bit size of desired precision for final result, $\log N$ — maximum possible modulus of fresh plaintext, $\log Pq$ — size of big integer modulus multiplied by fresh ciphertext modulus, $\log \Delta$ — size of scaling factor.

Remarks

In general, it is very good practice to immediately rescale after performing multiplications. This reduces both error and ciphertext size, and hence reduces computational and communication overhead in the grand scheme of things. In this work, the bulk of estimations and simulations were done assuming the needed level for multiplications around 13. Hence, as per table 3.1, the chosen parameters were $h = 64$, $Pq = 2^{930}$, $q = 2^{744}$ and $\Delta = 2^{49}$. In practice, N is not chosen, and in any case 2^{16} fully satisfies the needs for the operations performed in this work.

Finally, a touch on the weakness identified by Li and Micciancio [59]. Indistinguishability against CPA+ is only required if the data is shared with untrusted parties. To this end, Cheon, Hong and Kim [29] proposed a new decryption routine for sharing, $DecForShare_{sk}(ctxt, B_{ctxt})$ during which an additional noise is added to the decryption result:

$$m \coloneqq \langle ctxt, sk \rangle + e \pmod{q} \tag{3.15}$$

where $\langle \cdot \rangle$ denotes the original decryption procedure, and $e \leftarrow D_{\mathbb{Z}^n, B_{ctxt}}$ is an error sampled from a discrete Gaussian distribution over $\mathbb{Z}^n$ — n length of encoding vector — with upper bound B_{ctxt} w.r.t. maximal norm $|\cdot|_\infty$.

While the tight bound for B_{ctxt} is not fully derived theoretically yet, some implementations choose conservative bounds (such as HElib [50]), while others (such as SEAL [74]) advise against sharing decrypted ciphertexts and instead treating them as private information for the secret key owner only. In the particular case of this work, the considered approach is that given how the final decrypted value is assumed to be correlated with information pertaining to an individual's health status, it would be counter-intuitive for the secret key owner (i.e., the User in our construction) to share the decrypted values.

3.2. Network-Based Positioning Technologies

A very important concept for performing contact tracing for individuals positioned at integer locations is first being able to collect or estimate accurately the locations mobile devices. More specifically, there is a need for sub-metre location accuracy so that contact

tracing can be performed accurately in turn. To this end, the current 5G and the upcoming 6G show promising results. Technologies such as massive Multiple Input-Multiple Output (MIMO) localization, millimeter wave communication (mmWave), Ultra-Dense Networks (UDN) and Device-to-Device (D2D) communications will facilitate network-based localization in the near future.
In current 5G networks, there are a number of both conventional signal-processing and machine learning-based technologies that could potentially achieve sub-metre accuracy [65].
On the conventional side, there are a number of promising technologies which accurately manage to localize devices within the metre, using a number of algorithms and input data types, in both realistic and simulated environments. Said technologies are:

- Dynamic fingerprint matching algorithm paired with a received signal strength indicator in ultra-dense 5G networks for indoor localization (simulated) [82];
- Extended Kalman filter paired with angle of arrival input data in ultra-dense 5G networks for outdoor localization (realistic) [64];
- Extended Kalman filter paired with uplink reference signal input data in dense and ultra-dense 5G networks for outdoor localization (simulated) [57, 58];
- An expectation maximization algorithm and a subspace-spaced algorithm paired with uplink reference signal input data for indoor localization (simulated) [76];
- Unscented Kalman filter with time of arrival and angle of arrival input data for indoor localization (simulated) [54];
- Cramer-Rao bound derivation paired with time difference of arrival and angle of arrival input data for outdoor vehicle localization (realistic) [55];
- Deriving Ficher information of 5G and GNSS (Global National Satellite System) with simulated GNSS and 5G signals for outdoor localization (simulated) [16];
- Orthogonal frequency division multiple access (OFDMA)-based visible light communication positioning paired with light signals and received signal strength data for indoor localization [81].

Quite clearly, there is quite a number of enabling algorithms and technologies to accurately estimate end device location in 5G.
Machine learning, used in conjunction with massive MIMO and mmWave communications, currently seems to be a rather pretentious option, in the sense that location accuracy below 2m — both indoor and outdoor — is only achieved under very specific conditions: low

noise, high number of antennas or base stations, line-of-sight (LOS) propagation. On the other hand, in the presence of noise and non-line-of-sight propagation, error can grow to exceed 20m. Of the more promising approaches, we mention:

- Neural Networks paired with angle of arrival data for both indoor and outdoor localization [36];
- Densely-connected Neural Networks paired with received signal strength and GNSS signal for outdoor localization [56];
- Neural Networks paired with channel state information for both indoor and outdoor localization [40].

In 6G, the projections are that current solutions will be improved in terms of cost and efficiency, while new enabling technologies and applications will allow for a more fine-grained localization [21, 61, 62, 73, 77, 80]. For the purposes of this work, however, worst-case sub-metre accuracy will suffice.

4 | Privacy-preserving Network-based Contact Tracing Protocol

In this chapter, the contact tracing protocol is described. The protocol involves a government authority, mobile operators, and the respective mobile operators' subscribers, in order to generate privately a risk score for each user as a measure of their contact with other users and the other users' respective infection status. In section 4.1, general information about the protocol is given.
Section 4.2 formally describes the considered score computation algorithm, as well as the inter-entity communication phases.
The final section 4.3 provides details on how the privacy-preserving implementation was done.

4.1. Protocol Architecture

This section gives a high-level description of the protocol. Subsection 4.1.1 includes preambular details on the classes of entities involved — Users, Mobile Operators and Government Authority — such as the correspondence between real-life physical entities and the different actors in the protocol, as well as each of their respective responsibilities and roles. It has three subsections, one for each type of entity. Subsection 4.1.2 establishes the three major privacy goals of the protocol — confidentiality of location, confidentiality of contact, and confidentiality of status and risk score — in relation with whose interest it is to respect them. Moreover, it establishes the considered attack model.

4.1.1. Modelling of Involved Entities

The protocol considers three types of involved entities: Users, Mobile Operators (MO), Government Authority (GA). For convenience, every User is considered to be subscribed to

a single MO, and all Users are subscribed to some MO; moreover, every User is considered to have a single mobile device. Subscription to MOs hence defines a partition of the set of Users within a GA's jurisdiction.

Users

The Users are regular individuals/ subscribers, each identified one-to-one with their mobile devices (i.e., mobile device $i \equiv$ User i). Each User has an associated infection status, which can either be positive or negative. As they move, their current location updates accordingly. For contact tracing in the context of the protocol, Users each have a certain infection status — as a $1/0$ value — and current location — as a pair of coordinates (x, y) — as properties at a considered time t. Every User is assumed to have willingly and knowingly signed up to have their location and status used for the purposes of the protocol.
As their location updates according to their movement, relatively frequent estimations are done at the MO level (e.g., every 30 seconds), which in turn triggers the contact tracing phase calculations.
They can poll their own MO to receive their own risk score.

Mobile Operators

MOs are the providers of access to mobile services. They are assumed to have the capability to estimate their Users' locations within the metre, either through one of the techniques described in section 3.2 or through some other means. Regardless of location estimation techniques, the MO is thought to abide by 5G security standards ([11] or similar), and communications between parties are considered secure. For the purposes of the protocol, they collect their Users' locations, as well as receive their Users' status in order to perform the contact tracing computations in the protocol.
The protocol assumes communication phases occur periodically (Ticks), during which MOs exchange values of their Users' locations and infection status. In order to gain access to their Users' status, at each Tick count multiple of a larger Period including 0 (e.g., for minutely Ticks and daily Period, Tick count must be a multiple of 1440), the communication phase starts with the GA sending the Users' status to the MOs. After the information exchange is done, the contact tracing routine is performed for the current state of the system (i.e., User locations and status).

Government Authority

The GA holds the infection status of all users, as far as they are known by the authorities and the Users themselves. That is to say, if a User is confirmed positive, it is assumed that by the next Period, the GA will be aware of that. This is achieved through communicating with the appropriate medical and administrative authorities, given the Users' consent to make use of their data.
Distinctly from the MOs' communication Ticks, the GA communicates every Period, which is a multiple of the Ticks. This is done as a correspondence to how often countries reported their positive Covid cases during the pandemic (i.e., daily, on average). As a general line of thought, the Period was envisioned as equivalent to a day versus minutely Ticks in the MO, although that is in no way meant to be the only use case.
An alternate version of the protocol where the GA is made aware of whether the Users are at risk is described in [18, 19]. In this version, either the GA or the MO are considered trusted, which in part might break the confidentiality goals established in the next section.

4.1.2. Attacker Model

In order to define the attacker model, we first establish what the privacy goals are. The goals all relate to the general security concept of confidentiality — data is only accessible to authorized parties — and are as follows:

1. Location confidentiality. A User's location is only known to the user itself and their MO. The User has an interest in achieving this goal so as not to have their location data used without consent. The MO has an interest in achieving this goal so as not to have competitors or other potentially malicious parties use data collected by them;

2. Contact confidentiality. Information about whether a User is in contact with another User and the number of times they were in contact can only be known to the respective Users themselves. If the Users share the same MO, then the MO may be party to this information; if the Users are from different MOs, then any MO cannot know the identity of the other MO's User, nor can they know the number of times they were in contact. The User has an interest in achieving this goal so as not to have their contact data used without consent. The MOs have an interest in achieving this goal so as not to have their competitors or any other potentially malicious parties use data collected by them;

3. Infection status and score confidentiality. User infection status can only be known

to the User themselves and the GA. A User's risk assessment score is by default only known to themselves. The User has an interest in achieving this goal as it is highly sensitive data, certainly not to be used without their consent. The GA has an interest in achieving this goal so as not to lose the trust of the Users.

An attacker is therefore defined as any party — malicious or not — that would break the confidentiality goals. The communications themselves between entities via 5G are assumed to be secure under 5G standards, with correctly configured equipment. Hence the focus of this subsection will be on ensuring the privacy goals are met within the system, by the entities involved in the protocol. The MOs and GAs are to be considered honest but curious, as defined in [69]. To be more specific, it will be assumed that they would execute the protocol as instructed, but given the data and the opportunity, they would attempt to violate the privacy of the Users.

4.2. Score Computation

This section describes how the protocol is intended to work in the absence of the privacy requirements. User locations are considered to be pairs of (x, y) coordinates. Consider metaparameter δ to be the threshold of contact. That is, for contact tracing purposes, if two Users are within distance δ, then they will be considered in contact; otherwise, they will not. The following subsections define status encoding (in subsection 4.2.1), contact evaluation (in subsection 4.2.2), and finally risk score calculation (in subsection 4.2.3).

4.2.1. Infection Status

A user i's infection status at a given time t is encoded in a variable $status_i^{(t)}$, such that:

$$status_i^{(t)} = \begin{cases} 1, \text{ if } User_i \text{ is infected at time } t, \\ 0, \text{ otherwise.} \end{cases} \tag{4.1}$$

This is considered to be accurately known by the GA at every start of Period.

4.2.2. Contact

Contact between two Users is evaluated by comparing the distance between them with the contact distance threshold δ. If the distance is below δ, then the users are considered in contact. For users $User_i$ and $User_j$ at a given time t, having set a distance function between users $Dist : (User_1, User_2) \mapsto \mathbb{R}$, this can be modeled with a variable $contact_{ij}^{(t)}$,

such that:

$$contact_{ij}^{(t)} = \begin{cases} 1, \text{ if } Dist(User_i, User_j) \leq \delta, \\ 0, \text{ otherwise.} \end{cases} \tag{4.2}$$

The choice for the distance function warrants at least a short discussion. Depending on the setting, the *Dist* function can be Euclidean, Manhattan/taxicab, squared Euclidean etc. In a plaintext-only setting, using anything but Euclidean distance would at a first glance seem pretentious. In a setting that does not support root calculations, square or otherwise (e.g., if User locations would be represented as Shamir secret shares [75], or if encrypted under a fully-homomorphic scheme), [18, 19] proposed using the squared variant of Euclidean distance. Moreover, this *Dist* function is closely linked to the value of the contact threshold δ. To make this point clearer, if two Users are considered to be in contact if within, for example, 2m of one another, and if *Dist* is chosen to be squared Euclidean, then δ would have to be 4.

4.2.3. Risk Score

User risk scores are calculated by performing, for each respective User, the sum of the infection status of all their respective previous and current contacts. At a fixed time t_0:

$$score_i^{(t_0)} = \sum_{t \leq t_0} \sum_{j:contact_{ij}^{(t)}=1} status_j^{(t)}. \tag{4.3}$$

4.2.4. Inter-entity Communication

In order to facilitate the transfer and updating of information, a number of actions are performed periodically by the MOs and the GA. The frequency of periods differs from MOs to GA: the MOs are assumed to communicate with a relatively low period — e.g., minutely, every 30 seconds etc. — which shall be called a Tick from now on, while the GA communicates with a relatively high period — e.g., daily — which shall be called a Period.
During each Tick, the following are assumed to happen in order:

1. Each MO estimates all their Users' locations $(x_i, y_i), \forall User_i$;
2. Contact evaluation $contact_{ij}, \forall j \neq i$ for each User $User_i$ is performed by their respective MO;

3. User scores $score_i, \forall User_i$ are updated by their respective MO;

At every Period, the GA will communicate to each MO MO_i the contact status of all Users $User_i$ subscribed to the respective MO. Hence the Period, while not necessarily a multiple of the Tick, must first occur before the first Tick.
In addition to the periodic actions, there is an unscheduled entity-triggered events that can occur, namely the User-triggered score request. During this event, the User polls their MO for their current score, after which the MO sends the requested value to the User.
While the above description of the protocol is a bit too optimistic for a real-world scenario in terms of data transparency, it provides insight into how the protocol is intended to ultimately work and makes the privacy-preserving modifications easier to follow.

4.3. Privacy-Preserving Implementation

In the final version, the main change from the plaintext protocol is that information cannot flow in plaintext between unauthorized parties. Therefore, it has to be encrypted (or not to flow at all, but that would not be a sensible solution). As it is a main point of emphasis in this work, the chosen cryptographic system will be a fully-homomorphic one, more specifically CKKS [26]. Exact terminology and functions vary slightly depending on the particular cryptographic system used. In the following subsections, the fully-homomorphic encryption (FHE)-based protocol is described. In subsection 4.3.1, the building-block operations of a general FHE scheme are described at high level. Subsection 4.3.2 presents the necessary major changes to the ideal version described in section 4.2 to achieve a practical scheme. The changes include using a tessellation of the full protocol area, as well as a contact function that approximates comparisons without performing them, and finally the usage of CKKS as the underlying cryptographic system. The full form of the protocol is described in the final subsection, 4.3.3.

4.3.1. Cryptographic Primitives

This subsection describes at a high level the general capabilities to be expected from a fully-homomorphic cryptographic system. Denoting by $Enc(x)$ the encryption of value x, for all intents and purposes, this can be viewed at a very high level as an asymmetric encryption system that allows:

- An unlimited number of addition-like operations with ciphertexts — such as addi-

tions, subtractions — both with plaintexts and ciphertexts:

$$cipherAddition(Enc(x), Enc(y)) = Enc(x + y); \tag{4.4}$$

$$cipherSubtraction(Enc(x), Enc(y)) = Enc(x - y). \tag{4.5}$$

$$plainAdd(Enc(x), plain) = Enc(x + plain). \tag{4.6}$$

$$plainSub(Enc(x), plain) = Enc(x - plain). \tag{4.7}$$

- A limited number of multiplication-like operations with ciphertexts — multiplication of two ciphertexts, multiplication with a plaintext, exponentiation with plain integer exponent, multiplicative inversion (the possibility to perform the latter depends on the specific cryptographic system chosen; for example, CKKS [26] does not support it):

$$cipherMult(Enc(x), Enc(y)) = Enc(x \cdot y); \tag{4.8}$$

$$cipherExp(plain, Enc(x)) = Enc(x^{plain}); \tag{4.9}$$

$$plainMult(plain, Enc(x)) = Enc(plain \cdot x); \tag{4.10}$$

$$cipherInverse(Enc(x)) = Enc(x^{-1}). \tag{4.11}$$

- Essentially no other types of operations, unless additional manipulations are done to the encrypted values prior to encryption.

Under these circumstances, comparisons of encrypted values are highly discouraged — while splitting the plaintext into bits prior to encryption, encrypting each bit separately and then treating the created structure as such allows comparisons to be performed by using a series of logical operations, this greatly reduces both the number of potential multiplications that can be performed on the ciphertext, as well as the possible size of the plaintext; moreover, the computational complexity grows at a rate comparable to

exponential for each plaintext.

4.3.2. Adaptations from Plaintext Version

This subsection motivates and describes the manipulations done in order to make the protocol practical and secure. Quite obviously, an FHE scheme will be used to encrypt data and to perform computations on it. The necessary encryption procedures are described in sub-subsection 4.3.2. Moreover, a tessellation procedure is described for the protocol area in sub-subsection 4.3.2. In parallel, another case is made for using an approximate formula for the contact function in sub-subsection 4.3.2.

Tessellation

Pairwise comparing all Users' locations for possible contacts would be unrealistic (e.g., a User near PoliMI could surely not have a contact with a User near Duomo while they are in the respective locations), even considering plain locations. It is even more unrealistic to do so in cipher domain, as the computations are significantly slower in fully-homomorphic encryption schemes, mainly due to large ciphertext size. As a solution, a tessellation of the total protocol area is considered. Each area is assigned an ID $area\_ID$, while Users in the area will be associated with the $area\_ID$. During the information exchange between MOs at Ticks, the $area\_ID$ of each User is transmitted along with the rest of the information, and contact tracing will only be performed between Users within adjacent areas. While this tessellation need not be composed of equally-sized squares, every delimited sub-region can be included in such square. As a side note, not considering equal square tessellation areas seems to just complicate matters unnecessarily. From here onward, the tessellation of the area under protocol consideration will be considered as a splitting into equal-side squares.

Encryption Context

The encryption happens for the cases when information flows from authorised to unauthorised parties, as per the privacy goals set in section 4.1.2. Therefore User locations need to be encrypted when shared with any other party than the MO of the User itself, and User infection status need to be encrypted before the GA shares them with the MOs. These encryptions cause in turn the contact status between users of different MOs to be encrypted by default, and also the risk score be encrypted, as the operations performed would involve the encrypted infection status. Assume that each User $User_i$ has public key pk_i, and that Enc_pk is a function that encrypts plaintexts under public key pk. Then

the encryption of information

- Status encryption: assuming the status of $User_i$ at a given time t to be $status_i \in \{0, 1\}$, the GA encrypts it under some other User $User_j$'s public key pk_j, resulting in

$$es_{ij} = Enc_{pk_j}(status_i). \tag{4.12}$$

- Coordinate encryption: assuming the coordinates of $User_i$ at a given time t to be (x_i, y_i), $User_i$'s MO encrypts them under another User $User_j$'s public key pk_j, resulting in

$$(ex_{ij}, ey_{ij}) = (Enc_{pk_j}(x_i), Enc_{pk_j}(y_i)). \tag{4.13}$$

- Distance encryption: Assume $User_i$ is subscribed to some MO_m and $User_j$ is subscribed to $MO_n, n \neq m$. When MO_m calculates the squared Euclidean distance between $User_i$ and $User_j$, the coordinates of $User_i$ will be available to it in plain, while the coordinates of $User_j$ will be received encrypted under pk_i. The squared Euclidean distance becomes

$$\begin{aligned} Dist_{ij} = cipherAdd(&cipherMult(plainSub(ex_{ji}, x_i), plainSub(ex_{ji}, x_i)),\\ &cipherMult(plainSub(ey_{ji}, y_i), plainSub(ey_{ji}, y_i))). \end{aligned} \tag{4.14}$$

- Score update: Assume $User_i$'s current score is $score_i$, encrypted under pk_i, and that $contact_{ij}$ is the contact evaluation between $User_i$ and $User_j$ encrypted under pk_i. Then the operation for updating the score of $User_i$ becomes

$$score_i = cipherAdd(score_i, cipherMult(es_{ji}, contact_{ij})). \tag{4.15}$$

  Notably, if both $User_i$ and $User_j$ are subscribers of the same MO, $contact_{ij}$ can be calculated in plain, as it will only involve values accessible to the MO without encryption. That is, the formula becomes

$$score_i = cipherAdd(score_i, plainMult(es_{ji}, contact_{ij})). \tag{4.16}$$

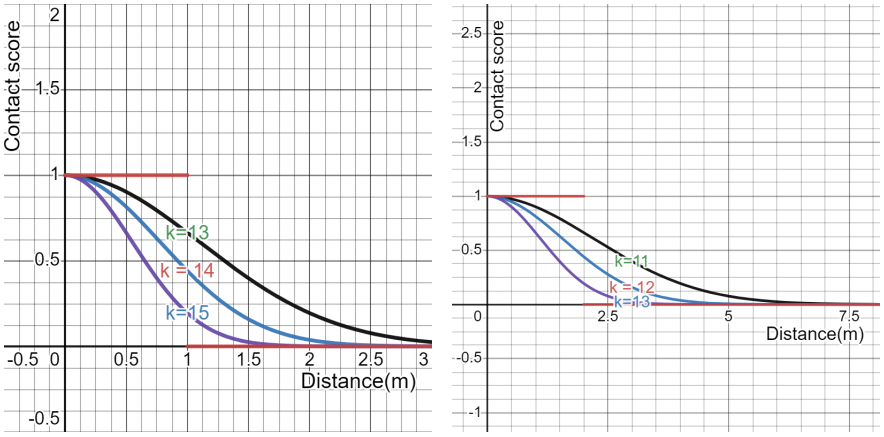
The contact formula calculation was not mentioned because it will be discussed in the very next sub-subsection.

Fully-Homomorphic Contact Formula

Accurate comparisons between numbers within CKKS is rather costly in terms of multiplications, especially at high numbers of polynomial degree — which is required to ensure 128 bits of security. Considering the tessellation in the protocol area proposed in subsubsection 4.3.2, an approximation formula is proposed to evaluate contact between two Users. Let l be the side of the squares used for tessellating. Then, between any two Users in adjacent areas, the maximum possible distance is $2\sqrt{2}l$. Leveraging this, at a given time t the new proposed contact formula between two Users — in plaintext form — using only additions and multiplications is

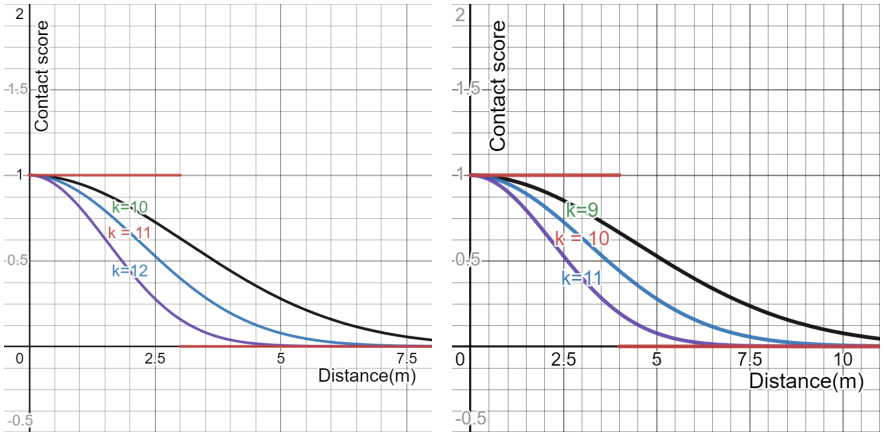
$$contact_{ij} = (1 - \frac{Dist(User_i, User_j)}{8l^2})^k, \tag{4.17}$$

where $Dist(User_i, User_j)$ is the squared Euclidean distance, and k is a suitably chosen exponent. The parameters for the formula are tessellation area side l and k. It can be assumed that tessellation area side is fixed, considering a trade-off between computational efficiency and privacy — the larger the area, the higher the computational cost; the smaller the area, the more likely an MO the User is unsubscribed to is to guess User locations. Considering the tessellation side fixed, the exponent k is chosen to be the highest power of 2 that best approximates the step function at the contact threshold δ. This is because, using repeated squaring, the level of the final ciphertext will be the same for any power between $2^(x-1)+1$ and 2^x inclusively. As a heuristic, the best power of 2 to use is the one that produces the closest value to $1 - \frac{1+sgn(x-\delta)}{2}$ at the point $x = \delta$. Such a choice yields a favorable balance between assigning higher weight to close contacts, while also not assigning weight to farther Users. The choice for exponent k changes with the chosen contact threshold. As shown in figures 4.1 and 4.2, the higher the considered contact threshold, the lower the exponent k that needs to be used to best approximate the contact formula. Within the figures, the red lines parallel to the x axis represent the step function that models the presence/absence of contacts depending on distance — 1 if the distance is under the considered threshold, and 0 if the distance is above.



(a) k choices for $\delta = 1, l = 50$ (b) k choices for $\delta = 2, l = 50$

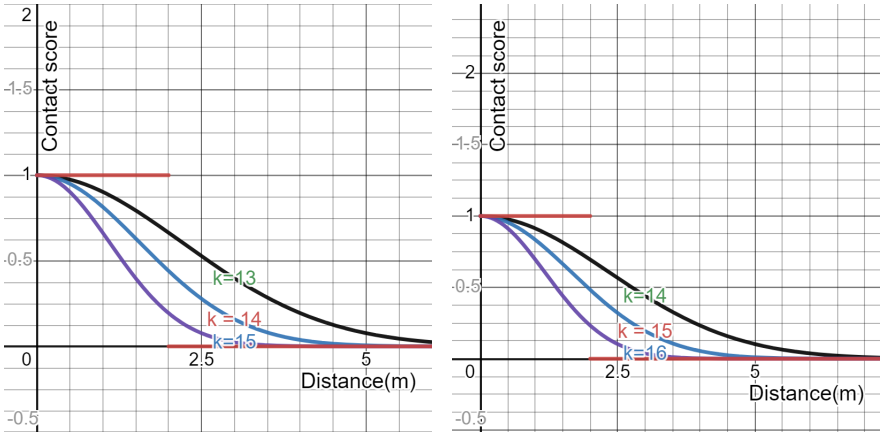
Figure 4.1: Contact score as function of inter-user distance with three choices of exponent for thresholds 1 and 2.



(a) k choices for $\delta = 3, l = 50$ (b) k choices for $\delta = 4, l = 50$

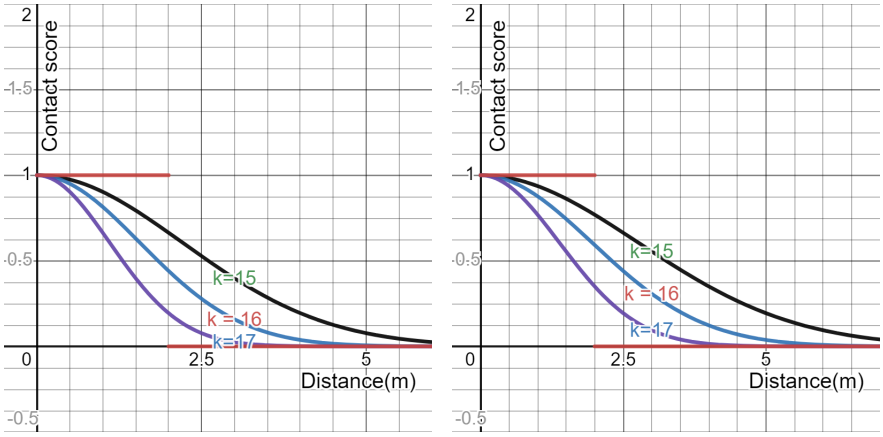
Figure 4.2: Contact score as function of inter-user distance with three choices of exponent for thresholds 3 and 4.

The choice of exponent also depends on the length of the sides of the tessellation areas. As shown in figures 4.3 and 4.4, the longer the side length l, the higher the need exponent k.



(a) k choices for $\delta = 2, l = 100$ (b) k choices for $\delta = 2, l = 150$

Figure 4.3: Contact score as function of inter-user distance with three choices of exponent for tessellation sides 100 and 150.



(a) k choices for $\delta = 2, l = 200$ (b) k choices for $\delta = 2, l = 250$

Figure 4.4: Contact score as function of inter-user distance with three choices of exponent for tessellation sides 200 and 250.

In cipher domain, the contact function becomes

$$contact_{ij} = cipherExp(k, plainAdd(1, plainMult(-\frac{1}{8l^2}, Dist(User_i, User_j)))). \quad (4.18)$$

where the exponentiation is done through repeated squaring.

4.3.3. Final Protocol

This subsection describes the final version of the contact tracing protocol, with specific CKKS [26] usage. It details all the phases of the protocol, from setup, Tick and Period communications, to the unscheduled User score requests. The opening sub-subsection lists the assumptions made in order to be able to run the protocol successfully.

Assumptions

This sub-subsection details the protocol assumptions. It describes the conventions which all entities should agree upon to have a smooth interaction, along with suppositions about the connections between them. Said conventions are related to common parameter selection, common formula selection and synchronized times. The final assumption is about communication being possible at any time required. The assumptions related to running the protocol are:

- All the entities have agreed on the same δ contact threshold (e.g., 2m), along with the respective contact formula. This may be softened to only having the MOs agree between one another, in accordance to the social-sanitary context. That is because δ is only involved — if indirectly — in the computation of the contact between Users, which is performed at the MO level.
- All the entities have agreed on the same CKKS parameters. That is to say, all entities encode complex vectors of the same length — which must be greater or equal to 128 — into polynomials of the same degree $poly\_deg$ (e.g., 256), using the same scaling factor sc_fact. All Users produce secret keys according to the same Hamming weight $h = polg\_deg/4$, along with the corresponding public keys. All fresh ciphertext moduli c_mod are the same, as well as all big integer P sizes for relinearization and respect the parameter settings for security against the latest known attacks (as per [31]).
- All entities agree on computing the necessary information using the same formulae and algorithms. More specifically, distance is evaluated as per formula 4.14, contact is evaluated as per formula 4.17, and User risk scores are updated as per formulae 4.15 and 4.16.
- The times at which entities communicate are fixed and known by all participants. It is additionally assumed that the clocks of all participants are synchronized, so that the Ticks and Periods commence at the same time for all.

Protocol Setup

This sub-subsection describes the actions to be performed before commencing the de facto communication between entities for the purposes of contact tracing.
During the setup phase, every User $User_i$ generates a suite of keys: a secret key sk_i, a public key pk_i and a relinearization key rk_i. The public key and rk_i is sent to the MO, sk_i is kept for the User only. The public key is also sent to the GA — but not the relinearization key.
After every MO has received the public keys of all its Users, a public key exchange commences. All MOs send the public keys of all their Users to all other MOs and hence access to all public keys is granted to all MOs. The User relinearization keys are not shared between the MOs. Moreover, all MOs initialize their Users' scores to $Enc_{pk_i}(0)$, for each $User_i$ with public key pk_i.
The GA collects the status of all Users from the relevant medical and administrative authorities. Then, after having received the public keys from all Users, proceeds to encrypt each User's status under all other Users' public keys.
After all of the above are performed by the entities, the protocol communication phases can begin. They are described in the next sub-subsection.

Communication Phases

The communication phases are split into three categories: Tick communications, Period communications, and unscheduled communications. The protocol commences via a Period communication, which is initiated by the GA and repeats with a relatively low frequency (e.g., daily). During this phase, the GA sends to each MO the status of their respective Users encrypted under all other Users' public keys. The Tick communications are initiated by User movement and location updates. These events then trigger the information to be exchanged between the MOs, after which the contact tracing and scoring computations are being performed by each respective MO. The unscheduled communications involve a User polling the MO for the score, after which the score encrypted under the User's public key is sent to the User. Then, the User decrypts the value using the secret key and finds the score in plain.

Period communication: to each MO MO_k, the GA sends

$$GA \rightarrow MO_k : [es_{ij} : \forall i \text{ s.t. } User_i \text{ subscribed to } MO_k, \forall j \neq i], \tag{4.19}$$

where es_{ij} is an encrypted status as defined in formula 4.12. The order of elements in the list corresponds to the order of the MO's Users' public keys, or the correspondence is made otherwise. The Period repeats in accordance to received updates from the medical and administrative authorities.

Tick communication: as the location of $User_i$ updates from (x_i, y_i) to (new_x_i, new_y_i), the update is estimated by the MO. Automatically, according to the tessellation area $User_i$ finds itself in after the update, the new area ID $area\_ID_i$ is assigned to $User_i$. Afterwards, the inter-MO communication begins. Each MO_m sends to every other MO_n a list of areas its Users are in.

$$MO_m \to MO_n : [area\_ID_i : \forall i \text{ s.t. } User_i \text{ subscribed to } MO_m]. \tag{4.20}$$

The order of elements in the list corresponds to the order of the MO's Users' public keys, or the correspondence is made otherwise. After this, each MO iterates through the list of its Users' areas, and for each one, encrypts the User's coordinates under the public keys of other MO's Users in adjacent areas. Adjacency is determined by the following criterion: if two areas touch at any point (i.e., sides or corners), then they are adjacent. Then it sends the encrypted data to the corresponding MO, along with the encrypted status of the User. The package sent between MOs is of form

$$\begin{aligned} MO_m \to MO_n : [(es_{ij}, (ex_{ij}, ey_{ij})) : \forall i, j \text{ s.t. } User_i \text{ subscribed to } MO_m \\ \text{and } User_j \text{ subscribed to } MO_n \text{ with } area\_ID_i \text{ adjacent to } area\_ID_j]. \end{aligned} \tag{4.21}$$

After all such packages are received by an MO, the contact tracing computations begin; no further communication between MOs is performed until the next Tick. Contact tracing computations are split into two categories: contact tracing between the MO's Users — can be done in plain — and between the MO's Users and other MOs' Users — must be done in cipher domain. The plain contact computations can be either performed via the threshold formula 4.2, or via the estimating formula 4.17. From here onward it will be assumed they are done via the estimating formula. The cipher contact computations are done using the formula 4.18. After the contacts between Users are evaluated, the scores are updated using formula 4.16 for Users of the same MO, and formula 4.15 for Users of different MOs respectively, for each User. This concludes the internal computations following the inter-MO communication phase.

Unscheduled communications occur when a User sends a request under the protocol to

the subscribed MO. This request is assumed to be the only User-to-MO communication possible after the setup phase. Upon receipt of the request, the MO sends back to the User its encrypted score, which the User can then decrypt to reveal the plain score. Figure 4.5 presents a sketch of protocol communication phases.

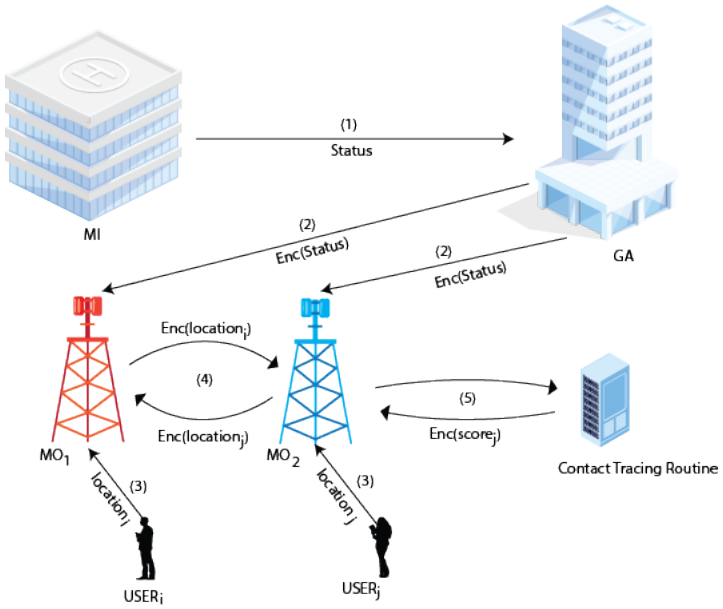


Figure 4.5: Sketch of Protocol Functionality.

Within the figure 4.5, the phases are presented in order of appearance. Phase (1) represents communication of User status between the relevant medical institutions and the GA. Phase (2) represents the communication of encrypted status to the respective MOs. Phase (3) represents the assessment of User locations by their respective MOs. Phase (4) represents the exchange between the MOs at each Tick. Phase (5) represents the contact tracing routine, ending with the update of encrypted User scores.

5 | Simulation and Results

This chapter describes the simulation context and results for the work. It begins by establishing a Gauss-Markov mobility model [25, 60] for User movement in section 5.1. Then, in Section 5.2, the metrics in view are described. The third Section 5.3 presents the simulation setup, in terms of used hardware and software construction. Fourth Section 5.4 describes and interprets the results. Finally, Section 5.5 draws general conclusions regarding the outcomes.

5.1. User Mobility Model

As large scale fine-grained mobility data is difficult to collect and even more difficult to find without hitting a number of paywalls, this work has resorted to producing synthetic mobility data. In order not to have too much computational load on producing mobility traces, the chosen approach was to use something relatively simple and quick, while also not being too simplistic. To this end, the Gauss-Markov model [25, 60] was chosen, as it allows variable velocity, angle of movement, as well as potential rest periods. The generator for the mobility traces was the pymobility Python library [68], more specifically the Gauss-Markov mobility generator model. Its implementation was inspired by the mobility model survey done by Camp and others [25].
In general, the aim was to simulate random movements of individuals randomly distributed over the protocol simulation area such that both average walking and speed-limit driving — considered 50 km/h or 14 m/s — were included. Hence, the parameters were set with a 50% chance to change direction and velocity, with a mean velocity of 7 m/s and variance 7. The result yields a iterator of a list of doubles (x, y) of size equal to the desired number of Users in the simulation context.

5.2. Metrics

This Section describes the considered metrics when producing simulation results. The security of the system was set through a no-compromise approach — i.e., parameters set

according to FHE security standards in relation to the latest known attacks as per [31]. Having established that, the considered metrics are as follows:

- *Error in score computation*: The considered error was a combination of the mean absolute error between threshold score in plain (as generated by using the formulae 4.2 and 4.3) and curve score encrypted (as generated by using the formulae 4.17, 4.15, and 4.16), as well as the threshold score in plain versus the curve score in plain. CKKS [26] adds error to fresh encryptions by default, and during every multiplication, the size of the error grows to a sum of the errors of the two ciphertexts. Moreover, the curve-based computation also adds errors compared to the threshold one. That happens because the weight assigned to every possible distance follows a smoother decrease that essentially assigns some weight to every distance up until $2\sqrt{2}l$, where l is the tessellation area side length. Aiming for full lack of error is infeasible, due to the two reasons mentioned above. The error in score computation metric therefore is a measure of how big the difference is between plain and encrypted curve computations, and the relation with the threshold score.

- *Computational complexity and run time*: The main source of complexity for the CKKS scheme — and all fully-homomorphic encryption schemes for that matter — is the number of multiplications to perform. This is especially true if the required security goals impose choosing large keys, along with large ciphertext moduli. For example, to reach the desired goal of $\lambda = 128$ bits of security (as is recommended for sensitive private data, such as User location and even more so infection status), one must choose a polynomial degree of at least 256, which in turn imposes a secret key size of 64 bits or more. Moreover, the need for a multiplicative level of at least 14 requires a ciphertext modulus of at least 930 bits, meaning that every multiplication between ciphertexts implies multiplications of two pairs of 256-entry vectors with 930 $\times$ 930 bit entries, followed by a costly relinearization operation. This metric shows the increase in computational complexity as the security of the system increases. Run time is a direct consequence of complexity, paired with the hardware available. However, it yields a reasonable palpable measure of how the system performs on a personal machine. It was found very instructive to show run time for every tick, and the impact various parameters have on it. Additionally, a graph of how ciphertext-ciphertext multiplication time decreases after every rescaling is reported.

- *Communication overhead*: Communication overhead is a measure of the amount of information exchanged by the different entities during the communication phases of the protocol. The communication overhead is reported as a function of full protocol area size, tessellation area size, User density and MO count. The main

generators of overhead are obviously the MOs, along with the GA. User overhead is relatively negligible and will not be measured. The overheads are reported as MO or GA overhead per User, as functions of tessellation area size and full protocol size, respectively.

5.3. Simulation Setup

The hardware used is a laptop with an Intel(R) Core(TM) i7-9750H processing unit running at 2.60GHz with 16GB RAM (15.8GB available). It runs a Windows 11 operating system, onto which a Windows Subsystem for Linux running Ubuntu 22.04.1 LTS with 8GB of allocated RAM memory. The simulation code is written in Python3, inside a Conda envirnoment configured to run py-fhe [8] and pymobility [68]. This Section describes the simulation settings and construction. It starts by listing simulation parameters and their meaning in subsection 5.3.1, and continues by describing the entity classes in the code construction and their roles in subsection 5.3.2.

5.3.1. Simulation Parameters

Before the simulation is run, the parameters are established. Said parameters can be split into two categories: simulation parameters pertaining to the issues not related to encryption, and encryption parameters pertaining to the encryption scheme. The simulation parameters are:

- Population size: integer POPSIZE; Represents the number of Users in the system, and also the number of entities simulated by the mobility generator;
- Total protocol area dimensions: (max_x, max_y) integer pair. Represents the dimensions of the protocol area, and also the total dimensions of the area considered by the mobility generator. During the simulations ran for generating results, it was considered that $max\_x = max\_y$;
- Tessellation area dimensions: $(area\_x, area\_y)$ integer pair. Represents the sizes of the tessellations. During the simulations ran for generating results, it was considered that $area\_x = area\_y$;
- Mobile operator count: integer MO_count. Represents the number of mobile operators executing the protocol exchanges over the protocol area. $MO\_count \geq 2$ for the protocol to make sense;
- Contact threshold: integer thr. Represents the distance under which the Users are

considered in contact. In general, during simulations, $thr = 2$ was considered;

- Infected count: integer INFECTED_COUNT. Represents the count of infected Users in the initial population. For the protocol to make sense, INFECTED_COUNT $\geq$ 1;
- Exponent: integer k. Represents the number of repeated squaring operations performed during contact calculation. It is determined by the tessellation area sizes, as well as the contact threshold.

The encryption parameters are the scaling factor, polynomial degree, ciphertext modulus and big number modulus of the CKKS encryption scheme. They are wrapped by an instance of the CKKSParams class in py-fhe.

5.3.2. Controller and Entity Classes

Simulations are created by running a controller-style class that keeps track of the passing of time and of the movements and additionally triggers all the calls to the appropriate methods inside the entity classes. It is constructed from a movements_iterable object (essentially an iterator created from an instance of a pymobility iterator of POPSIZE nodes over a (max_x, max_y) area that is sampled every 60 iterations to mimic minutely movements), a params object (an instance of the py-fhe CKKSParams class), MO_count, and two tuples — $(area\_x, area\_y)$ and (max_x, max_y). At initialization, the Controller does the following in order:

1. It iterates once through movements_iterable and rounds the output locations to the nearest integer coordinates;
2. It sets the current time to 0;
3. It builds a GA object;
4. It creates MO_count MO objects with incremental IDs and links them with each another and with the GA;
5. It creates POPSIZE User objects with incremental IDs and links each of them to the MO with ID $User\_ID \ \% \ MO\_count$ and with the GA, where % denotes modular reduction. This ensures an even distribution of Users to MOs. Each User with ID $User\_ID$ is initialized with a location equal to the $User\_ID$'th entry in the Controller's current locations, which is passed to their respective MO.

The status of the Users is only stored inside the GA, in plaintext. It is set by randomly choosing INFECTED_COUNT positions in a POPSIZE vector of bits and assigning

the value 1 to them; the rest are 0. The resulting vector is stored in the GA as the $User\_status$ vector.
There are three families of protocol entities used for the simulations: two plaintext families and an encryption-based family. By family it is meant the Controller, GA, MO and User classes. The plaintext families are the threshold family — which uses the contact formula based on threshold evaluation 4.2 — and the curve family, which use the contact formula 4.17 based on the approximations created for ciphertext evaluation. The Controller for the plaintext families does not take encryption parameters as an initialization argument. Instead, the Controller for the threshold scheme takes a thr argument as described in subsection 5.3.1, while both the other two types of Controller take exponent k as an initialization parameter. The encryption-based family is essentially the encryption variant of the curve family.
At this point, the initialization phases for the plaintext families is over. For the encryption-based family, the initialization phase ends after the User initialization, during which encryption keys are generated and exchanged:

1. Upon creation, every User produces a key suite, of which he sends to the respective MO the public and the relinearization keys. In parallel, the User sends the public key to the GA;
2. The MO, upon reception of the keys, stores the relinearization key and relays the public key to all other MOs;
3. In parallel, the MO initializes the score assigned to the User to an encryption of 0;
4. The GA, upon reception of the keys, encrypts the $User\_status$ vector position by position under all the other Users' public keys.

After the initialization phase is completed, the Controller commences the communication phases as described in subsection 4.3.3. This is done via a major routine called Tick. During the Tick, the following happen:

1. Check if the current Tick count is a multiple of a Period. If so, the Period communications are started from the GA;
2. Tick communications are initiated in the MO objects;
3. New locations are generated from the movements_iterable and rounded to the nearest integers;
4. Tick count is incremented;
5. Locations are updated in each User, corresponding to the locations generated above.

The Period in the GA implies sending a list of encrypted status to each MO for each User as described in the Communication Phases sub-subsection 4.3.3, more specifically by formula 4.19.

The Tick in the MO triggers the following:

1. User locations are updated. The area ID assigned to each User is updated according to the location. The set of User references assigned to each area ID is updated accordingly;
2. A routine for evaluating the scores of its own Users. Contact is evaluated via the plain version of the approximate contact formula 4.17. The score of each User is updated through the score update formula for same-MO Users 4.16;
3. User locations are encrypted under the other MOs' Users' public keys;
4. Lists of areas, encrypted status and locations are exchanged with each respective MO as defined in the Tick Communications sub-subsection of 4.3.3.
5. After the exchanges are performed with all MOs, contacts and scores are evaluated preserving privacy. Squared distance between plain own User locations and encrypted locations is calculated, then multiplied with the encoded constant $-1/(8area\_x^2)$, subtracted from the encoded constant 1, and squared repeatedly according to the exponent k to produce the contact value in encrypted form. Then, the contact value is multiplied with the encrypted status as defined in the formula 4.12 and added to the User's score. This is performed for every User and concludes the Tick.

5.3.3. Simulations

This subsection describes the simulations set up for evaluating the performance of the construction. They are of two types: simulations that evaluate encryption system performance and simulations that evaluate the performance of the protocol as a whole.
The simulations that evaluate the encryption system performance imply running a number of tests to evaluate running time for atomic operations on the specified hardware. The first subtype of simulation evaluates run time for encoding, encryption, multiplications with plaintext and ciphertext, additions with plaintext and ciphertext, decryption, rescaling, and modulus lowering. The second subtype of simulation evaluates the decrease in time required to perform subsequent squarings after rescaling the same ciphertext. They are both performed outside of the protocol construction, over just an instance of the CKKS suite with variable parameters.

The second type of simulation involves running the protocol using the Controller class and all the classes created by it. The simulations involve running an instance of a Controller of some family against either a Controller of another family or Controller of the two other families. These simulations are enabled through the creation of iterator wrappers over the pymobility generator that copy the results of the original minutely iterator for two or three consecutive iterations respectively. In this way, the mobility data fed into each Controller in the same simulation is exactly the same. During the simulations, final scores are registered and compared in the different settings. In simulations involving the encrypted Controller, run time is also registered.

5.4. Illustrative Numerical Results

This Section describes the specific simulations run, and the obtained results, along with relevant figures and data tables.

Operation Execution Times

The considered simulation runs a selection of separate operations related to the CKKS scheme on a large number of random integers (of values up to 3700) and records the time it takes to perform each operation. The operations considered are the ones used in the implementation of the protocol. The results are then aggregated in a data table and the mean values and percentiles are reported. The considered simulations run on instances of CKKS with the same polynomial degree (256), ciphertext modulus (2^{744}), and big integer modulus (2^{930}). One simulation (see figure 5.1) is set up for a scaling factor of 2^{42}, while the other (see figure 5.2) is set up for a scaling factor of 2^{49}.

	encode	encrypt	cipher_mult	plain_add	decrypt	rescale	cipher_add	plain_mult	lower_mod
count	2481.000000	2481.000000	2481.000000	2481.000000	2481.000000	2481.000000	2481.000000	2481.000000	2481.000000
mean	0.001812	0.576493	1.817958	0.000211	0.285286	0.000313	0.000394	0.574050	0.000390
std	0.000814	0.155154	0.492322	0.000077	0.082430	0.000110	0.000133	0.158593	0.000126
min	0.001080	0.449983	1.411983	0.000150	0.219279	0.000228	0.000287	0.446846	0.000295
25%	0.001228	0.488102	1.538701	0.000171	0.239898	0.000256	0.000319	0.485300	0.000318
50%	0.001649	0.521831	1.644727	0.000185	0.256334	0.000277	0.000345	0.516507	0.000345
75%	0.001913	0.578563	1.825116	0.000209	0.284631	0.000310	0.000389	0.575717	0.000388
max	0.010405	1.526302	5.721932	0.001821	1.265971	0.002108	0.001701	1.941181	0.001928

Figure 5.1: Single operation times for scaling factor 2^{42}.

	encode	encrypt	cipher_mult	plain_add	decrypt	rescale	cipher_add	plain_mult	lower_mod
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	0.001178	0.465577	1.464089	0.000175	0.230166	0.000258	0.000321	0.466298	0.000325
std	0.000155	0.018331	0.056706	0.000030	0.012759	0.000034	0.000041	0.019490	0.000046
min	0.001075	0.447680	1.408827	0.000144	0.218799	0.000221	0.000278	0.358345	0.000292
25%	0.001115	0.455446	1.434243	0.000163	0.223805	0.000244	0.000302	0.456094	0.000305
50%	0.001139	0.460424	1.448314	0.000166	0.227003	0.000247	0.000307	0.460631	0.000310
75%	0.001185	0.467775	1.474176	0.000171	0.231694	0.000261	0.000323	0.469866	0.000325
max	0.003717	0.627894	2.220146	0.000460	0.335794	0.000712	0.000698	0.623772	0.000679

Figure 5.2: Single operation times for scaling factor 2^{49}.

The data collected shows that the most costly operation is the multiplication between ciphertexts (listed in the cipher_mult column of the tables). It is roughly three times more costly than the next two costliest operations, encryption and multiplication with plaintexts. Moreover, as multiplications between ciphertexts is performed repeatedly during each contact computation between Users of different MOs, it easily can be chosen as the bottleneck of the protocol in terms of computation. An interesting result is the decrease in multiplication time from scaling factor 2^{42} to scaling factor 2^{49}. As a general rule of thumb, it is preferred to use the highest scaling factor (that still provides security as per [31], illustrated in figure 3.1) possible to ensure the highest possible precision of the final result, as well as the highest decrease of ciphertext size during the rescaling procedure (which happens after every multiplication). The achieved results also show that fresh ciphertexts with higher scaling factors multiply quicker, which represents an added encouragement to adopt such a parameter setting strategy.

Repeated Squaring Times

The considered simulation runs produces an instance of the CKKS suite with the same parameters considered as the de facto security standard, namely polynomial degree 256, ciphertext modulus 2^{744}, big integer modulus 2^{930}, and scaling factor 2^{49}. Then, a pair of random integer coordinates is generated. The first one has $x, y \in [0, 50)$, while the second has $x, y \in [-50, 100)$ to emulate two Users in adjacent tessellation areas. Then a value equal to $1 - Dist/20000$ is encoded an encrypted, where $Dist$ is the squared distance between the coordinates — equivalent to the base of the exponentiation performed during the contact calculation. The resulting value is then repeatedly squared and rescaled by the scaling factor and the times of each subsequent multiplication are reported. This operation is repeated 13 times, equivalent to the number of encrypted multiplications and rescalings needed to run the contact tracing and scoring routines of the protocol

considering tessellation area side $l = 50$ and contact threshold $\delta = 2$. The observed multiplication times are reported in figure 5.3.

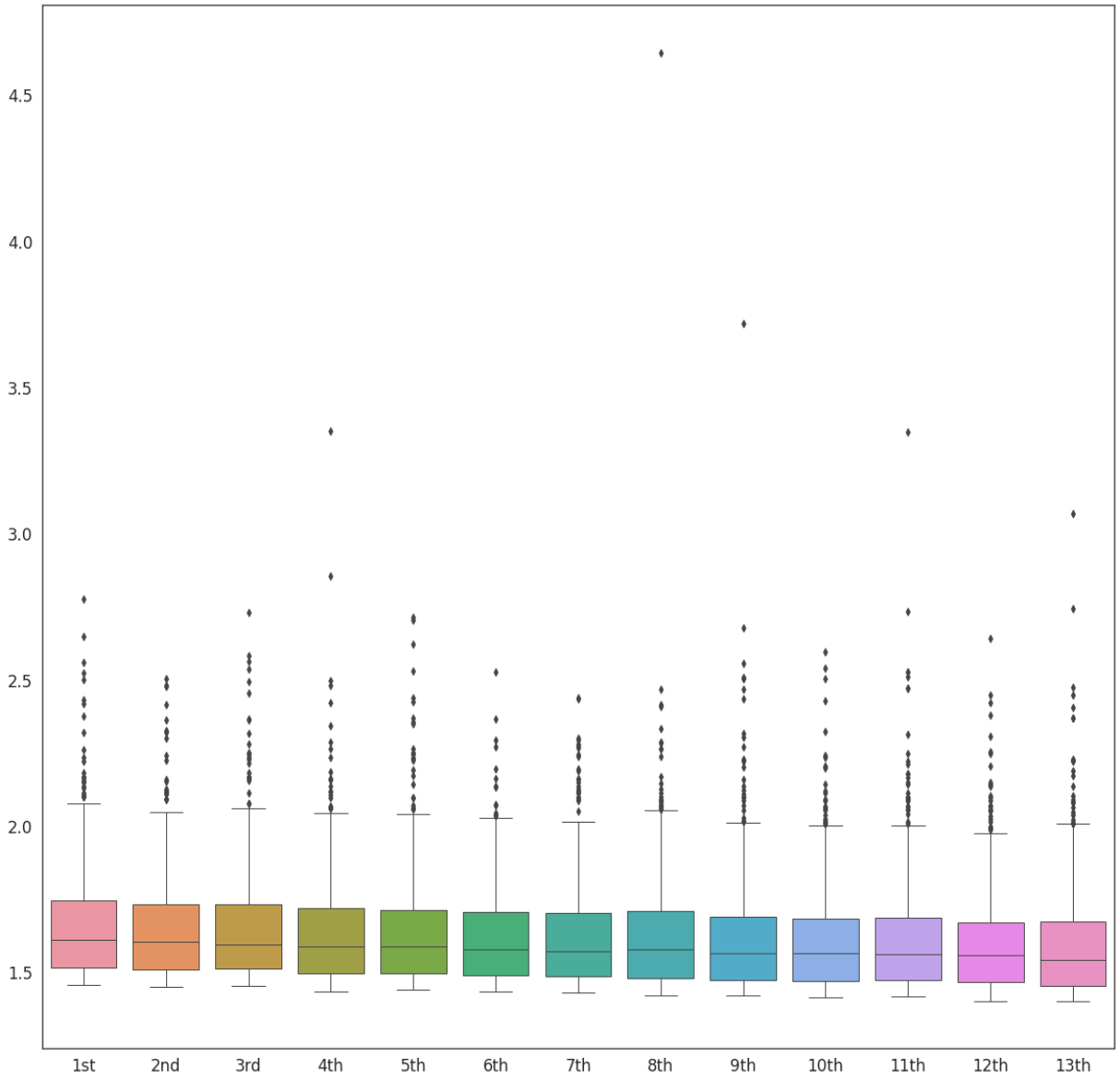


Figure 5.3: Multiplication times for decreasing ciphertext size visualization.

As can be seen in the figure, the time to perform a multiplication decreases as ciphertext size decreases, and this is correlated with the size of ciphertext modulus after rescaling. However, the decrease in ciphertext size does not fully account for the multiplicative complexity, as the relinearization complexity also depends on the big integer modulus, whose size remains unaffected by the rescaling procedure. The multiplication times between ciphertexts decrease from a mean of 1.63s for fresh ciphertexts (level 0) to 1.56s for level-13 ciphertexts. The exact numbers are described in figure 5.4

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th	13th
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	1.632639	1.624330	1.621779	1.613688	1.610978	1.604558	1.599756	1.601638	1.589846	1.583393	1.585215	1.577064	1.564906
std	0.158082	0.153335	0.153705	0.154523	0.158113	0.150872	0.156602	0.171039	0.162427	0.158022	0.160260	0.151398	0.151334
min	1.413795	1.408172	1.404101	1.402687	1.398438	1.395488	1.387704	1.386390	1.381550	1.369885	1.372969	1.363800	1.350798
25%	1.492826	1.487423	1.484518	1.477652	1.472501	1.469227	1.465674	1.461338	1.451145	1.447185	1.447386	1.441556	1.432078
50%	1.644350	1.634333	1.630338	1.623756	1.616773	1.614086	1.603297	1.605492	1.595608	1.590058	1.591293	1.585847	1.572538
75%	1.725777	1.715592	1.713979	1.704920	1.701695	1.694556	1.689632	1.686907	1.675260	1.671792	1.673719	1.663118	1.652192
max	3.024875	2.537769	2.731613	3.350518	2.847596	3.177957	3.094397	4.646902	3.720142	3.304663	3.348599	2.643640	3.069471

Figure 5.4: Multiplication times for decreasing ciphertext size description.

In addition to this, an estimate of how long a single contact score calculation takes, as well as how long the score update procedure takes was done and is reported in table 5.1.

Execution Times for Contact Tracing and Scoring

	Contact	Score
mean	19.177152	20.742058
std	1.637950	1.768309
min	16.796706	18.159914
25%	17.605163	19.049309
50%	19.516737	21.105453
75%	20.216645	21.852071
max	28.899934	30.812420

Table 5.1: Execution times for contact tracing and scoring (3000 runs).

Communication Overhead

The fact that relevant information is sent between the parties only once and that the calculations can be safely performed locally implies that one of the main advantages of the proposed protocol is reduced communication overhead. Indeed, on a per-User basis, on every run of the protocol (Tick), every MO produces a more than reasonable expected overhead. The GA overhead per User, while significantly larger, is only produced once per day, and hence can also be considered reasonable.

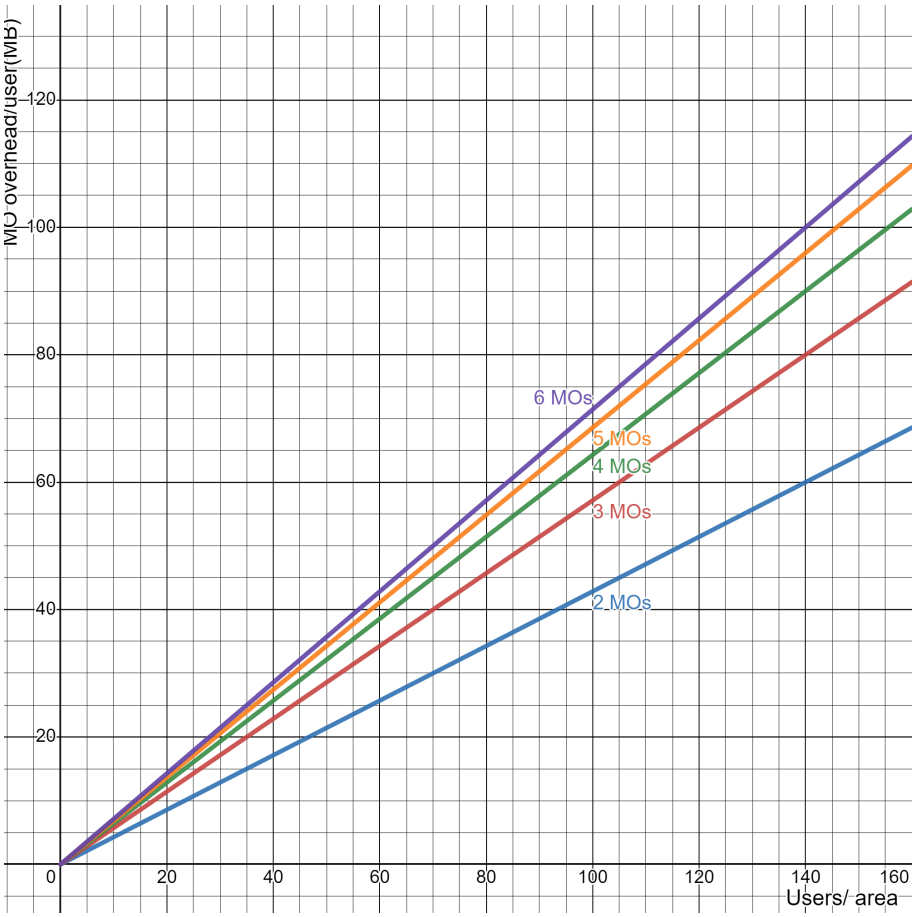


Figure 5.5: MO communication overhead per User as function of Users per area.

Figures show the mean expected overhead generated by each User for their respective MOs. In figure 5.5, the dependence of the average overhead generated per User by each MO is shown as a function of User density per tessellation area. The multiple lines in the figure represent the overhead generated when two to six MOs are participating in the protocol exchange. The tessellation area size considered is fixed at $50m \times 50m$. The estimations show an average of 8.142MB of overhead for two MOs for a Milan-level density of area (18.75 Users/$2500m^2$ as per [6]), while the average can grow to 77.138MB for a Manila-level density of area (around 108 Users/$2500m^2$ as per [1]). The relationship between User density and overhead per User is linear.

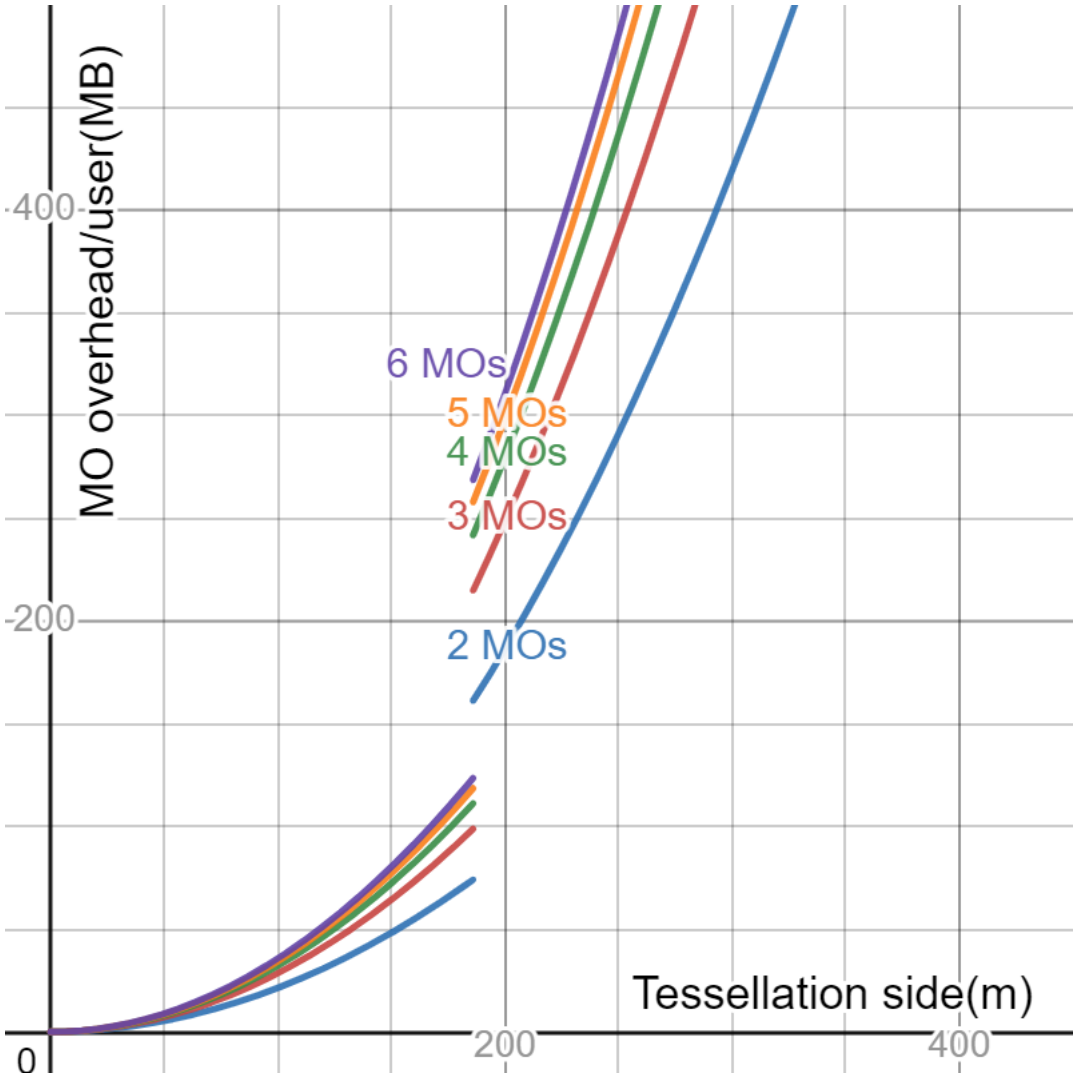


Figure 5.6: MO communication overhead per User as function of tessellation area side length.

Figure 5.6 shows the relationship between tessellation area size and overhead generated by each User, with different curves for different MO counts (from two to six). It considers a constant User density of $5000/km^2$, or $0.005/m^2$, which is equivalent to a Milan-level city considering 66% of the population has 5G-capable devices and all are subscribed to MOs participating in the protocol. The break in the graph at 186m is because the multiplicative level required to perform contact tracing according to the approximation formula 4.17 implies the need to pass from a ciphertext modulus of 2^{744} to one of 2^{1618}. The average communication overhead for a 50m-tessellation area side length is 5.357MB for two MOs, while it grows to 123MB per User for 6 MOs for the lower bound at 186m. At the tessellation area side length of 463m, the overhead generated for each User per Tick surpasses 1GB, even for the two MO case. The relationship between tessellation

area side and generated overhead per User is quadratic.

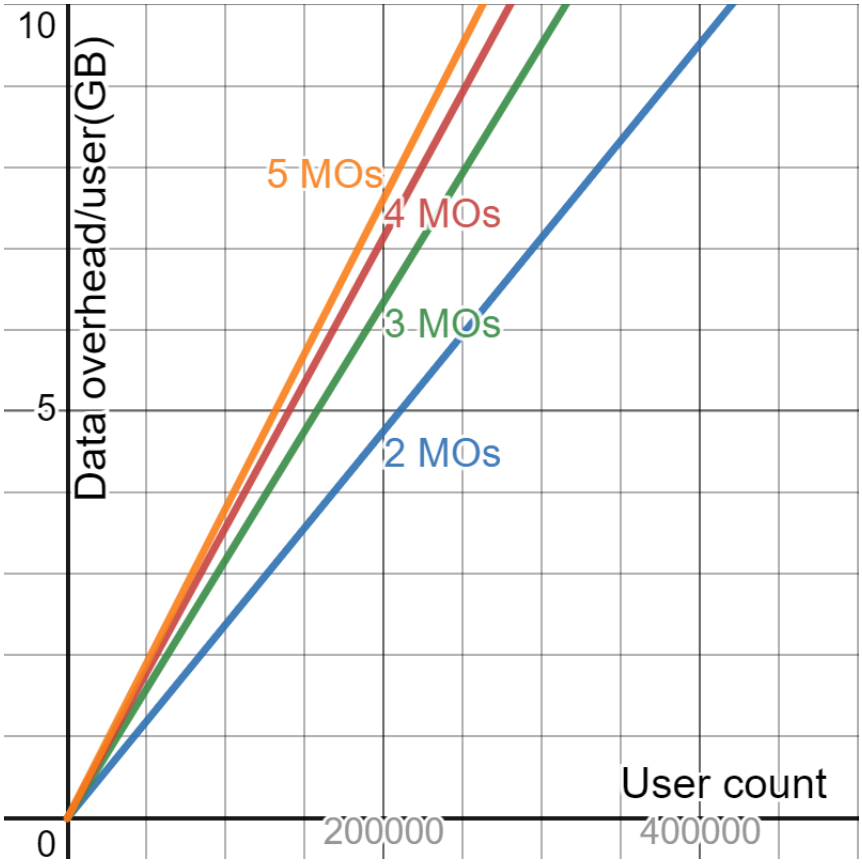


Figure 5.7: GA communication overhead per User as function of total User count.

The overhead generated by the GA per User is reported as a function of the total area under the action of the protocol — in figure 5.8, and as a function of the total number of Users in figure 5.7. The overhead is estimated for the protocol using CKKS with the polynomial degree set to 256 and ciphertext modulus 744 bits (i.e., a tessellation area of side length less than 186m). The considered area under protocol action is considered to be square-shaped, with a User density of $5000/km^2$. The relationship between GA overhead per User and total User count is a linear one, with 2.381GB of data generated for 100000 Users for two MOs. The relationship between GA overhead per User and protocol area side length is quadratic, with 2.076GB generated for each User on a $25km^2$ total area. This is considered acceptable, given that this overhead is generated with a low frequency (considered daily) relative to the frequency of protocol rounds.

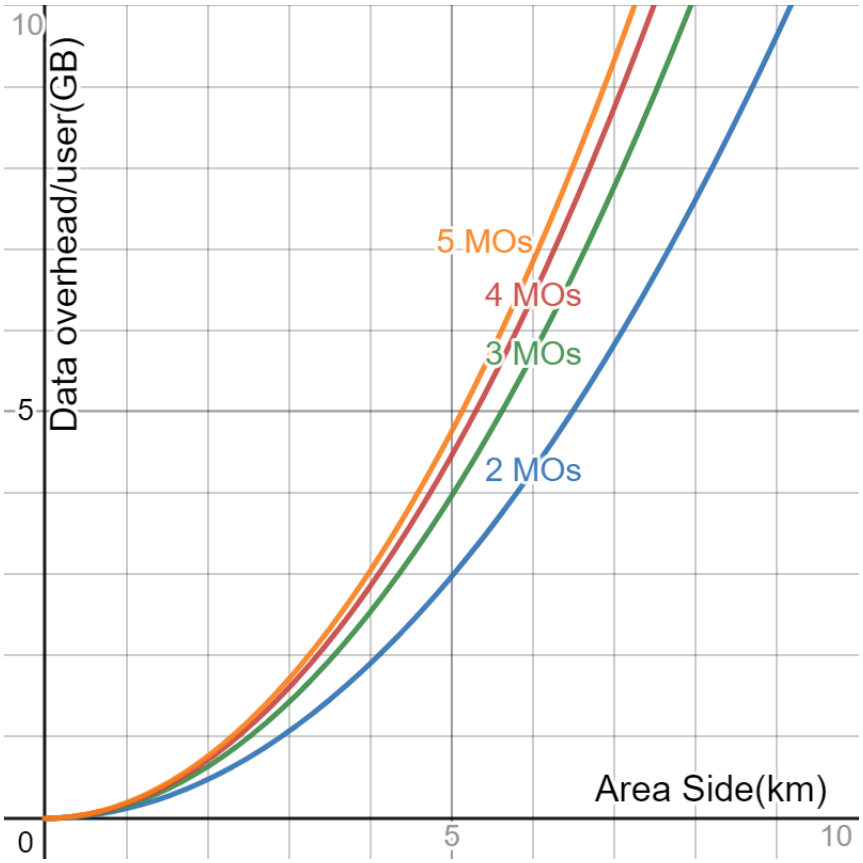
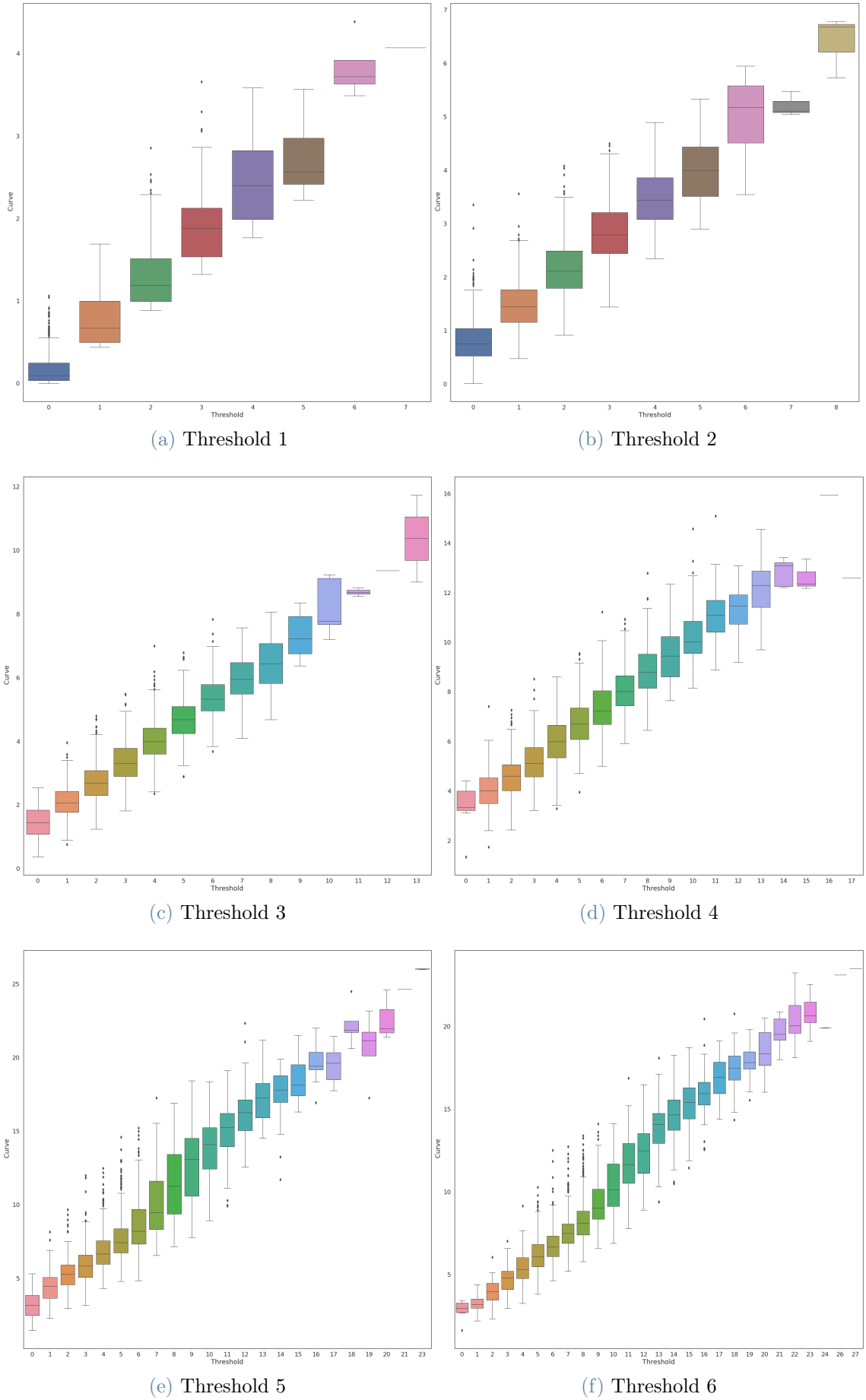


Figure 5.8: MO communication overhead per User as function of total area side length.

Threshold versus curve contact scores

The results in this subsection were obtained by running simulations of the plaintext Controllers against one another.

For each contact threshold δ from 1m to 10m, the corresponding exponent for the curve contact approximation formula was established — considering tessellation area side equal to 50m, heuristically the exponent k that produces the closest curve to the point $((\delta, 0.5))$ — and generate a threshold Controller with threshold δ, as well as a curve Controller with exponent k. All the simulations were run over a population of 500 of which 50 were infected, on an area of 200m $\times$ 200m, with a tessellation of 50m $\times$ 50m and two MOs. Depending on the value of δ, they were run against one another for a number of Ticks. For the initial run, all the pairs were run for 100 Ticks. Afterwards, the lower threshold Controller pairs were run for a higher number of Ticks, to achieve higher scores, while the higher threshold controller pairs were run for a lower number of Ticks to achieve the lower scores. All data was collected in pairwise files. The resulting data points were used to create the boxplots in figures 5.9 and 5.10 respectively — i.e., the coloured parts



(a) Threshold 1

(b) Threshold 2

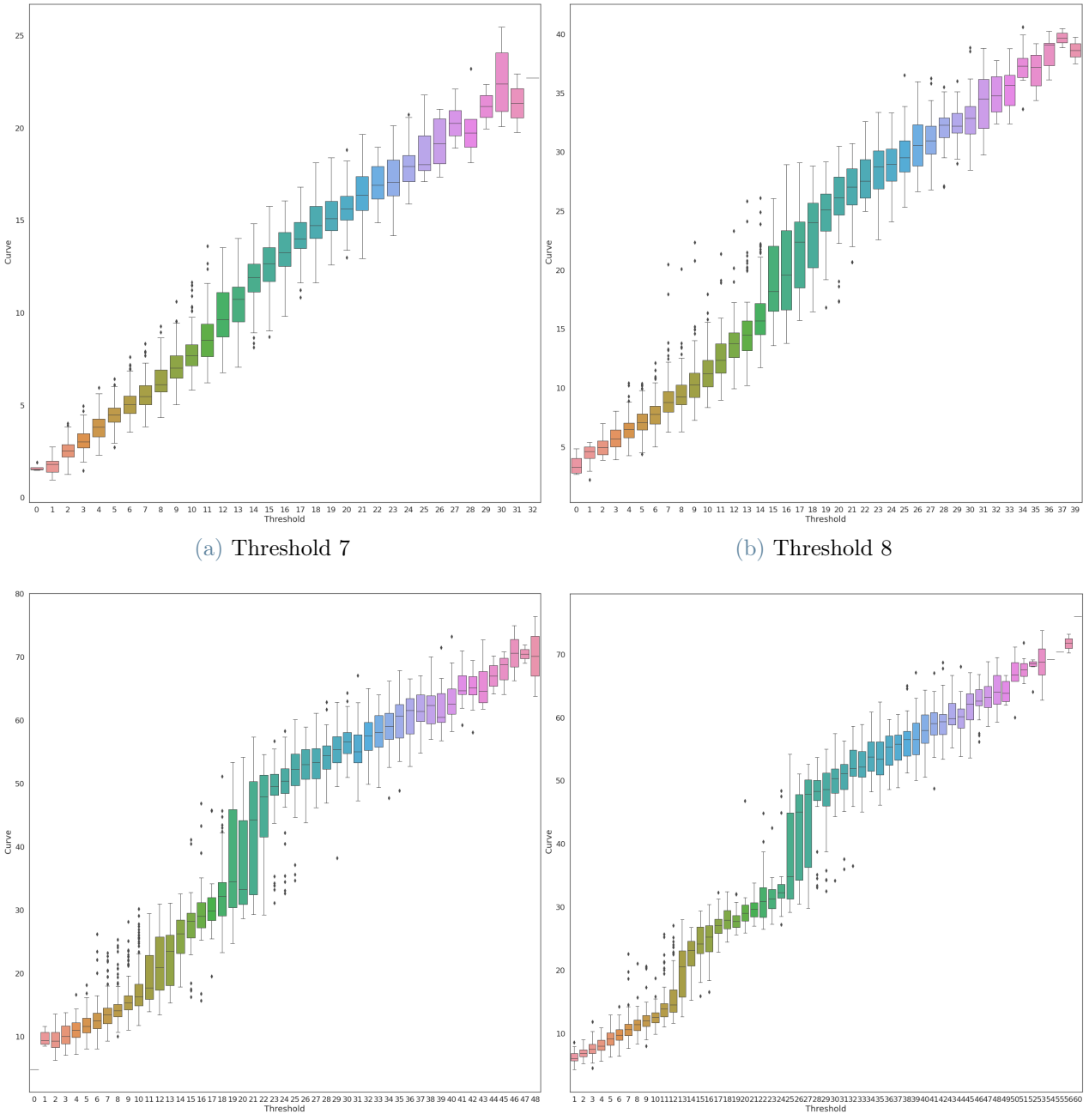
(c) Threshold 3

(d) Threshold 4

(e) Threshold 5

(f) Threshold 6

Figure 5.9: Comparison of threshold score versus curve score for thresholds 1–6



(a) Threshold 7

(b) Threshold 8

(c) Threshold 9

(d) Threshold 10

Figure 5.10: Comparison of threshold score versus curve score for thresholds 7–10

show the values between the 25th and 75th percentiles, the dots are considered outliers, and the remaining range of values is represented by the "whiskers". There is an obvious correlated increase in the curve score as the threshold score increases. However, as the threshold increases, the approximation formula 4.17 produces poorer approximations of the desired threshold contact evaluation procedure. Moreover, extreme threshold score values are rather poorly represented by the corresponding contact scores, with either missing values in some cases or lower threshold scores having higher corresponding contact scores. The scheme was mainly designed with contact threshold 2 in mind (as during the Covid pandemic, authorities around the world suggested keeping a distance of 1–2 metres between persons), and it performs quite well at low thresholds — the respective interquartile ranges almost partitions the space of values. For more overlapping value ranges, an equivalent score could be computed by taking the average of the corresponding threshold scores. At this point, this is an idea and possible grounds for future work.

Error in computation

For the assessment of errors in computation, the simulations performed involved running a plaintext curve Controller against an encryption Controller. The simulations were meant to evaluate the extent to which the CKKS cryptographic system-added errors affect the final results of the computations. Both Controllers are initialized with the same mobility iterator and the distinctive feature between different runs of the simulation is the encryption parameters used. To this extent, the main resource used for setting parameters was the table provided in [31], recreated in table 3.1. Generally, error depends on the scaling factor used, as well as on the multiplicative depth of the algorithms used. The following describe the errors obtained during simulations with the preferred parameters for system security (polynomial degree 256, ciphertext size 744 bit, big integer size 930 bit, scaling factor 49). On a simulation (two runs, one value was removed because the plaintext result was over 1) over a protocol area of 100m $\times$ 100m, using a tessellation of 50m $\times$ 50m for 20 Users of which 5 are infected, the absolute error behaves as described in table 5.2.

Absolute error plaintext curve vs ciphertext curve

	Error
mean	5.151382e-02
std	1.618113e-01
min	7.061351e-13
25%	5.350806e-12
50%	1.620244e-06
75%	6.200350e-04
max	6.639255e-01

Table 5.2: Absolute error plaintext curve vs ciphertext curve for 20 Users in 100m × 100m (2 runs of 10 Ticks each); one major outlier was removed.

Similarly, in a situation with the same encryption parameters, but with 10 Users over a protocol area of 150m × 150m, the error behaves as described in figure **??**.

Absolute error plaintext curve vs ciphertext curve

	Error
mean	5.436674e-06
std	1.206621e-05
min	6.572359e-12
25%	1.366804e-11
50%	2.125079e-11
75%	4.560400e-08
max	3.525788e-05

Table 5.3: Absolute error plaintext curve vs ciphertext curve for 10 Users in 150m × 150m (1 run of 10 Ticks).

The most relevant result is the max obtained error, as this will dictate the final error upon addition in cipher domain. In a setup with a sparse User distribution, the errors are lower than in denser situations.

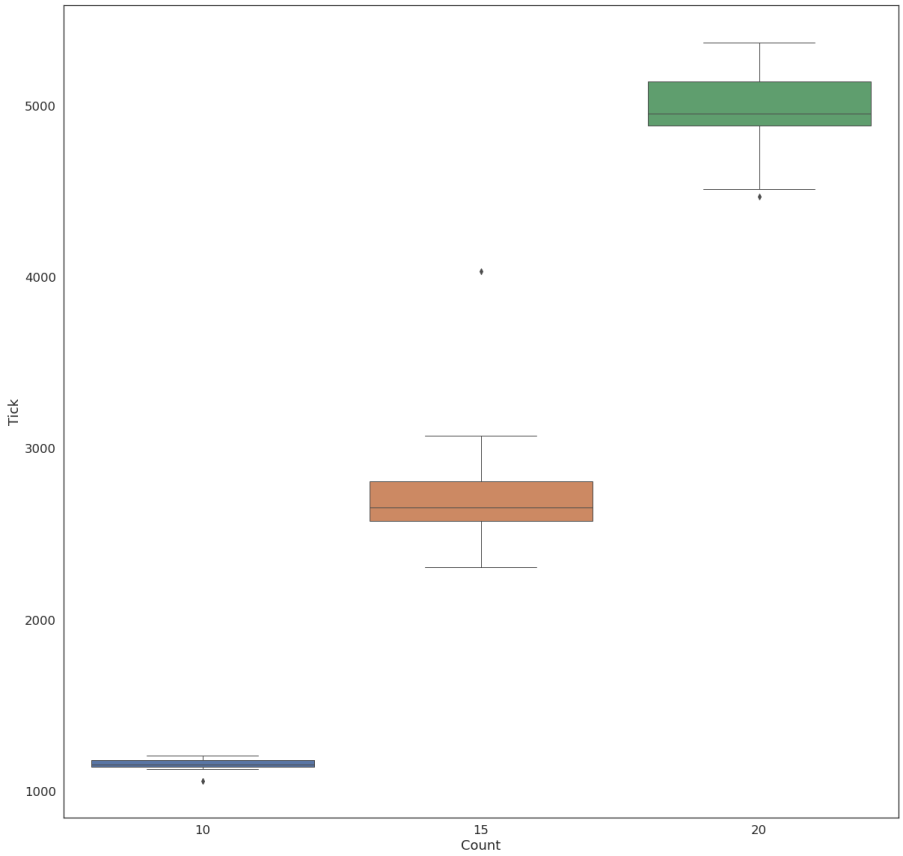


Figure 5.11: Seconds to perform an average Tick for User count.

Protocol round execution time

The main drawback of using fully-homomorphic encryption in general is the long computation times due to the multiplication of big ciphertexts — and additionally expensive relinearization operations. This is the reason why the simulations comparing encrypted calculations with plaintext ones are small in size compared to the relatively bigger simulations comparing plaintext Controllers. The expected computation times are quadratic in User count per tessellation area — similar to overhead. The first Tick time is expected to take longer than the following ones, as it also involves the respective encryptions at the GA level. Using polynomial degree 256, ciphertext modulus 2^{744}, big integer modulus 2^{930}, and scaling factor 2^{49} the dependence of Tick times on the population size is reported in figures 5.11 and 5.12. Figure 5.12 shows the distribution of the time it takes to perform the first Tick as a function of User count, while 5.11 shows the times taken to perform subsequent Ticks. The times were produced while running a series of simulations of the encryption Controller for 10 Ticks each. The 10 and 15-User simulations were run 5 times each, while the 20-User simulation was run 2 times. We recall an average time of around 18.5 minutes per Tick for 10 Users, 43 minutes for 15 Users, and 83 minutes for 20 Users.

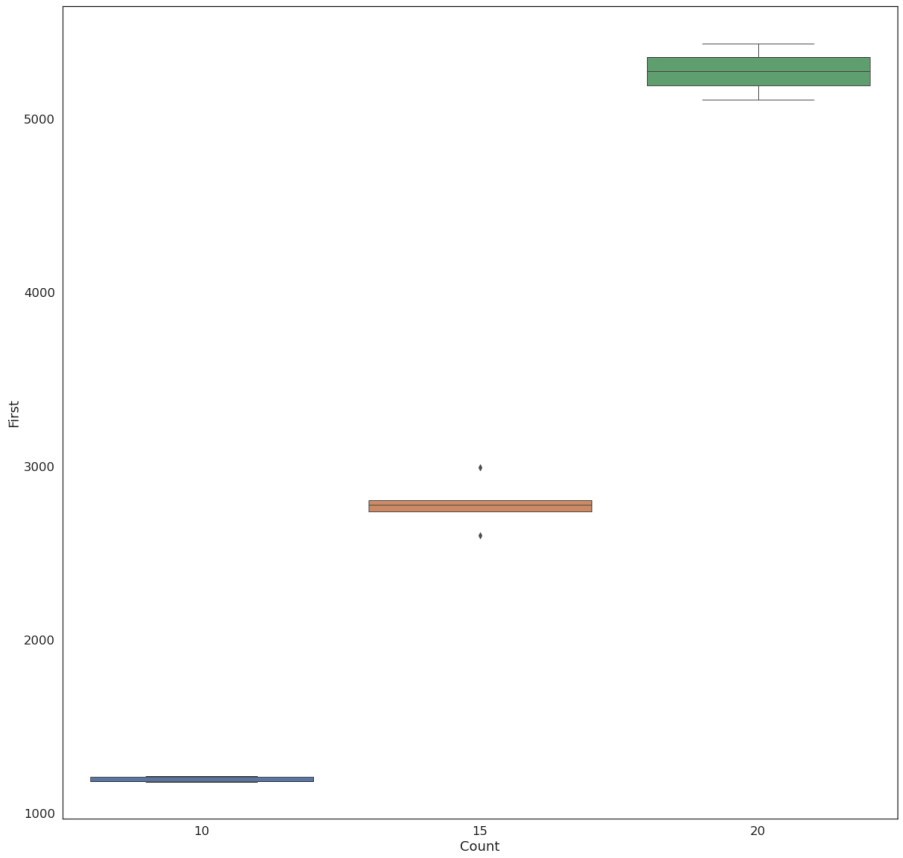


Figure 5.12: Seconds to perform the first Tick for User count

5.5. Summary

This summary describes the obtained results and draws conclusions regarding the performance of the system.

MO communication overhead per User depends on the used parameters — namely, ciphertext modulus — as well as the number of Users in a given area. In general, with tessellation area growth, ciphertext modulus grows, as well as the number of Users in the area. For only growing User densities over tessellation areas, the overhead grows linearly with User count. We recall overhead values can reach as low as 5MB for a city of Milan density.

GA communication overhead per User depends on the total number of Users affected by the protocol. The apparently unappealing overhead order of magnitude is softened by the fact that GA communication occurs at least 1440 times less often that the MO overhead — considering daily GA communications and minutely MO communications. For example, for a population of a million, considering 2 MOs, the overhead generated by the GA per User corresponds to roughly 17MB for each MO communication round.

All things considered, the resulting overhead is quite low.

Errors in computation occur two-fold. Once due to the approximations, and once due to the intrinsic errors added by CKKS. For parameters that provide a good security level (i.e., polynomial degree 256, ciphertext modulus 2^{744}, scaling factor 2^{49}), the resulting absolute errors due to CKKS are in the worst case 0.6634, with a median of order 10^{-8}. The differences in scoring due to usage of a different formula we report in the threshold figures. In general, the lower the threshold, the better the separation between approximate values corresponding to threshold scores at the cost of more multiplications and hence higher computational costs. On the other hand, the more the considered contact threshold grows, the lower the needed exponent for formula 4.17, while values are more difficult to classify.

Lastly, long computational times are the expected drawback of using FHE. The reported times range from 18 minutes per Tick for 10 Users distributed to two MOs to 83 minutes for 20 Users. This is rather slow. As the high running times are due to the size of ciphertexts, reducing the ciphertext size would imply either a loss in security — for smaller Hamming weights — a loss in potential multiplication level — for smaller ciphertext moduli — or both. The dependence of computational time on User count is roughly quadratic, as both the comparison operations are done in sequence by the simulation. We present as more relevant when considering high-scale performance the time taken to perform a single multiplication, as well as the average contact and score computation. For secure parameters, they are 1.632639s on average for a single multiplication, 19.177152s

on average to evaluate contact, and 20.742058 to compute score.

6 | Conclusion and Future Work

6.1. Conclusion

While the Covid pandemic may be mostly behind us as a society, new epidemics and contagions arise in all matter of corners of the world. In this work, a new approach to performing contact tracing was proposed and prototyped. Individuals identified with their devices have their locations estimated by their network operators and through the collaboration of certain government agencies, a privacy-preserving contact tracing and risk evaluation method is performed.

The advantages of using network operators for performing such a task are ease of access and the lack of need for direct interaction with an application or the added discomfort of having the users' batteries drained by keeping various transceivers on at all times, such as Bluetooth or GPS. All this stems from the operators' estimations of their users' locations and the possible sub-metre accuracy achievable in dense 5G environments.

Moreover, state of art fully-homomorphic encryption schemes provide an exceptional means of performing computations privately. This allows for reduced overhead while exchanging data between parties, as the necessary data need only be sent once.

An additional novelty of this approach is the approximation of a step function through a smooth addition and multiplication-based formula depending on a tessellation of the considered area of effect of the protocol and the distance within which individuals are considered in contact.

The main drawback of our solution is represented by the high computational time. Nevertheless, continuous improvements of FHE, both on the theoretical and implementation levels, promise to lower these high computational times to more acceptable values.

6.2. Future Work

The use of fully-homomorphic encryption provides an avenue of new and exciting ways of expanding the work done in this thesis.

6.2.1. Parallelism and Distribution

The protocol in this thesis was produced with single-thread execution in view, and provides a baseline for capabilities and performance. Additional concurrent execution can be performed in many places within the actual FHE operations, as well as within the calculations done for each contact tracing procedure, and not finally, different User contact computations and subsequent score updates can be executed by multiple threads, allowing a significant speed-up of what currently looks like a rather slow process.
Additionally, computations can be outsourced to potential third parties, to lighten the load currently placed upon wholly on the network operators. Of course, this will imply additional communication overhead, as well as a new potential point of breach for privacy concerns.

6.2.2. Algorithms and Alternate Schemes

While the approximation formula used in this protocol is clever and gets the job done, there are claims of better performance comparisons within the literature. Papers such as [30], [33] suggest that improvements can be done to the current scheme to make it more efficient in terms of execution times. They either exploit underlying number theoretical properties of the number fields considered in the construction of the encryption suite, or use alternative packing techniques yet to be explored in this work.
Moreover, there are a number of other schemes and implementations that have as of yet been unexplored and warrant attention for the future. The BGV [24], BFV [39], and TFHE [34] schemes, as well as the more ample usage of the RNS (Residue Number System) variant of CKKS [27] can potentially speed up the comparisons which make up the bulk of the computations in this work.
Not lastly, the full extent of the batching capabilities of CKKS has not been exploited in the prototype produced for this work. While such an approach would not reduce the data overhead generated by the government agents or by each round of the protocol, it can potentially reduce the frequency of rounds without incurring significant additional computational costs.

6.2.3. Advances in Hardware

The projected advances in specialized hardware, may allow these computations to perform much better in a future that is not that far away.
Upcoming software-defined architectures such as Cornami's Trustream [2] claim 10^6 quicker operations on FHE-based ciphertexts, quick enough for real-time computations on the en-

crypted data. Once the availability of such hardware will not be a matter of discussion, their impact on the current construction can be revisited and .

Bibliography

[1] Wikipedia list of cities proper by population density. https://en.wikipedia.org/wiki/List_of_cities_proper_by_population_density.

[2] Cornami website. https://cornami.com/trustream.

[3] Duality website. https://dualitytech.com/product/.

[4] Enveil website. https://www.enveil.com/products.

[5] Inpher XOR website. https://inpher.io/xor-secret-computing/.

[6] Wikipedia page on Milan. https://en.wikipedia.org/wiki/Milan.

[7] PALISADE website. https://palisade-crypto.org/.

[8] py-fhe repository (release 1.0). https://github.com/sarojaerabelli/py-fhe.

[9] Zama concrete documentation. https://docs.zama.ai/concrete.

[10] Py-SEAL repository (release 2.2). https://github.com/Lab41/PySEAL, July 2019.

[11] 5g cybersecurity standards — enisa, 2022. URL https://www.enisa.europa.eu/publications/5g-cybersecurity-standards.

[12] HEAAN (release 2.1). https://github.com/snucrypto/HEAAN/releases, Jan. 2022.

[13] Lattigo (release 4.0). https://github.com/tuneinsight/lattigo, Oct. 2022.

[14] PALISADE repository (release 1.11.2). https://gitlab.com/palisade/palisade-development, Sept. 2022.

[15] Zama concrete repository (release 0.2.0). https://github.com/zama-ai/concrete, Oct. 2022.

[16] Z. Abu-Shaban, G. Seco-Granados, C. R. Benson, and H. Wymeersch. Performance analysis for autonomous vehicle 5g-assisted positioning in gnss-challenged environments. *2020 IEEE/ION Position, Location and Navigation Symposium,*

PLANS 2020, pages 996–1003, 4 2020. doi: 10.48550/arxiv.2004.07380. URL https://arxiv.org/abs/2004.07380v1.

[17] D. Andreoletti, S. Giordano, G. Verticale, and M. Tornatore. Discovering the geographic distribution of live videos' users: A privacy-preserving approach. *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*, pages 1–6, 2018. URL https://ieeexplore.ieee.org/document/8647639/.

[18] D. Andreoletti, O. Ayoub, S. Giordano, M. Tornatore, and G. Verticale. Privacy-preserving multi-operator contact tracing for early detection of covid19 contagions. pages 1–6. IEEE, 12 2020. ISBN 978-1-7281-7307-8. doi: 10.1109/GCWkshps50303.2020.9367403. URL https://ieeexplore.ieee.org/document/9367403/.

[19] D. Andreoletti, O. Ayoub, S. Giordano, G. Verticale, and M. Tornatore. Network-based contact tracing for detection of covid-19 contagions: A privacy-preserving approach. *IEEE Communications Magazine*, 59:42–48, 9 2021. ISSN 0163-6804. doi: 10.1109/MCOM.001.2100015. URL https://ieeexplore.ieee.org/document/9566510/.

[20] A. A. Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, I. Quah, Y. Polyakov, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca. Openfhe: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915, 2022. URL https://eprint.iacr.org/2022/915. https://eprint.iacr.org/2022/915.

[21] A. Bourdoux, A. N. Barreto, B. van Liempd, C. de Lima, D. Dardari, D. Belot, E.-S. Lohan, G. Seco-Granados, H. Sarieddeen, H. Wymeersch, J. Suutala, J. Saloranta, M. Guillaud, M. Isomursu, M. Valkama, M. R. K. Aziz, R. Berkvens, T. Sanguanpuak, T. Svensson, and Y. Miao. 6g white paper on localization and sensing. 6 2020. doi: 10.48550/arxiv.2006.01779. URL https://arxiv.org/abs/2006.01779v1.

[22] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7417 LNCS:868–886, 2012. ISSN 03029743. doi: 10.1007/978-3-642-32009-5_50/COVER. URL https://link.springer.com/chapter/10.1007/978-3-642-32009-5_50.

[23] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 97–106, 2011. ISSN 02725428. doi: 10.1109/FOCS.2011.12.

[24] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. Fully homomorphic encryption without bootstrapping. 2011.

[25] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2:483–502, 2002. URL https://onlinelibrary.wiley.com/doi/10.1002/wcm.72.

[26] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. 2016.

[27] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. A full rns variant of approximate homomorphic encryption. *Selected areas in cryptography : ... annual international workshop, SAC ... proceedings. SAC (Conference)*, 11349:347, 2018. ISSN 16113349. doi: 10.1007/978-3-030-10970-7_16. URL /pmc/articles/PMC8048025//pmc/articles/PMC8048025/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC8048025/.

[28] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. Bootstrapping for approximate homomorphic encryption. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10820 LNCS:360–384, 2018. ISSN 16113349. doi: 10.1007/978-3-319-78381-9_14/FIGURES/3. URL https://link.springer.com/chapter/10.1007/978-3-319-78381-9_14.

[29] J. H. Cheon, S. Hong, and D. Kim. Remark on the security of ckks scheme in practice. 2020.

[30] J. H. Cheon, D. Kim, and D. Kim. Efficient homomorphic comparison methods with optimal complexity. In S. Moriai and H. Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 221–256, Cham, 2020. Springer International Publishing. ISBN 978-3-030-64834-3.

[31] J. H. Cheon, Y. Son, and D. Yhee. Practical fhe parameters against lattice attacks. *J. Korean Math. Soc*, 59:35–51, 2022. ISSN 2234-3008. doi: 10.4134/JKMS.j200650. URL https://doi.org/10.4134/JKMS.j200650.

[32] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10031 LNCS:3–33, 2016. ISSN 16113349. doi: 10.1007/978-3-662-53887-6_1/FIGURES/1. URL https://link.springer.com/chapter/10.1007/978-3-662-53887-6_1.

[33] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Improving tfhe: faster packed homomorphic operations and efficient circuit bootstrapping. 2017.

[34] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Tfhe: Fast fully homomorphic encryption over the torus. 2018.

[35] H. Cho, D. Ippolito, and Y. W. Yu. Contact tracing mobile apps for COVID-19: Privacy considerations and related trade-offs. 2020. URL `http://arxiv.org/abs/2003.11511`.

[36] M. Z. Comiter, M. B. Crouse, and H. T. Kung. A data-driven approach to localization for high frequency wireless mobile networks. *2017 IEEE Global Communications Conference, GLOBECOM 2017 - Proceedings*, 2018-January:1–7, 7 2017. doi: 10.1109/GLOCOM.2017.8254732.

[37] M. V. Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6110 LNCS:24–43, 2010. ISSN 03029743. doi: 10.1007/978-3-642-13190-5_2/COVER. URL `https://link.springer.com/chapter/10.1007/978-3-642-13190-5_2`.

[38] L. Ducas and D. Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9056:617–640, 2015. ISSN 16113349. doi: 10.1007/978-3-662-46800-5_24/COVER. URL `https://link.springer.com/chapter/10.1007/978-3-662-46800-5_24`.

[39] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

[40] K. Gao, H. Wang, H. Lv, and W. Liu. Toward 5g nr high-precision indoor positioning via channel frequency response: A new paradigm and dataset generation method. *IEEE Journal on Selected Areas in Communications*, 40:2233–2247, 7 2022. ISSN 15580008. doi: 10.1109/JSAC.2022.3157397.

[41] C. Gentry. A fully homomorphic encryption scheme. 2009.

[42] C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. *Cryptology ePrint Archive*, 2010.

[43] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and*

Lecture Notes in Bioinformatics), 8042 LNCS:75–92, 2013. ISSN 03029743. doi: 10.1007/978-3-642-40041-4_5/COVER. URL https://link.springer.com/chapter/10.1007/978-3-642-40041-4_5.

[44] Google. Covid-19_apps, . URL https://docs.google.com/spreadsheets/d/1qfbhFZbWCXd4GspSD6LL8CNmgzKEYtL-gQQk1gEKqrk/edit#gid=0.

[45] Google. Covid tracing tracker, . URL https://docs.google.com/spreadsheets/d/1ATalASO8KtZMx__zJREoOvFh0nmB-sAqJ1-CjVRSCOw/edit#gid=0.

[46] Google. Digital rights tracker supporting data, . URL https://docs.google.com/spreadsheets/d/1enCBRLVCo2Dp2B0AB3tEYvLc279i5LUuoGCzoelz8aQ/edit#gid=1023364174.

[47] Google and Apple. Exposure notification bluetooth® specification. 4 2020.

[48] Google and Apple. Exposure notification cryptography specification. 4 2020.

[49] S. Gorantala, R. Springer, S. Purser-Haskell, W. Lam, R. Wilson, A. Ali, E. P. Astor, I. Zukerman, S. Ruth, C. Dibak, P. Schoppmann, S. Kulankhina, A. Forget, D. Marn, C. Tew, R. Misoczki, B. Guillen, X. Ye, D. Kraft, D. Desfontaines, A. Krishnamurthy, M. Guevara, M. Perera, Y. Sushko, and B. Gipson. A general purpose transpiler for fully homomorphic encryption. 2021.

[50] S. Halevi and V. Shoup. Design and implementation of helib: a homomorphic encryption library. *Cryptology ePrint Archive*, 2020. URL https://github.com/homenc/HElib.

[51] E. Hernández-Orallo, C. T. Calafate, J. C. Cano, and P. Manzoni. Evaluating the effectiveness of covid-19 bluetooth-based smartphone contact tracing applications. *Applied Sciences 2020, Vol. 10, Page 7113*, 10:7113, 10 2020. ISSN 2076-3417. doi: 10.3390/APP10207113. URL https://www.mdpi.com/2076-3417/10/20/7113/htmhttps://www.mdpi.com/2076-3417/10/20/7113.

[52] R. Hinch. Effective configurations of a digital contact tracing app: A report to NHSX. *Christophe Fraser*, 1:1, 2020. URL https://www.pepp-pt.org. Alternative URL (11.04.22): https://cdn.theconversation.com/static_files/files/1009/Report_-_Effective_App_Configurations.pdf?1587531217.

[53] J. Hoffstein, J. Pipher, and J. H. Silverman. Ntru: A ring-based public key cryptosystem. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1423:267–288, 1998. ISSN

16113349. doi: 10.1007/BFB0054868/COVER. URL https://link.springer.com/chapter/10.1007/BFb0054868.

[54] Y. Jia, H. Tian, S. Fan, and B. Liu. Motion feature and millimeter wave multi-path aoa-toa based 3d indoor positioning. *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, 2018-September, 12 2018. doi: 10.1109/PIMRC.2018.8580805.

[55] A. Kakkavas, M. H. C. Garcia, R. A. Stirling-Gallacher, and J. A. Nossek. Multi-array 5g v2v relative positioning: Performance bounds. *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*, 2018. doi: 10.1109/GLOCOM.2018.8647812.

[56] R. Klus, J. Talvitie, and M. Valkama. Neural network fingerprinting and gnss data fusion for improved localization in 5g. *2021 International Conference on Localization and GNSS, ICL-GNSS 2021 - Proceedings*, 6 2021. doi: 10.1109/ICL-GNSS51451.2021.9452245.

[57] M. Koivisto, M. Costa, J. Werner, K. Heiska, J. Talvitie, K. Leppanen, V. Koivunen, and M. Valkama. Joint device positioning and clock synchronization in 5G ultra-dense networks. *IEEE Transactions on Wireless Communications*, 16(5):2866–2881, 2017. URL http://ieeexplore.ieee.org/document/7880669/.

[58] M. Koivisto, A. Hakkarainen, M. Costa, K. Leppanen, and M. Valkama. Continuous device positioning and synchronization in 5g dense networks with skewed clocks. *IEEE Workshop on Signal Processing Advances in Wireless Communications, SPAWC*, 2017-July:1–5, 12 2017. doi: 10.1109/SPAWC.2017.8227741.

[59] B. Li and D. Micciancio. On the security of homomorphic encryption on approximate numbers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12696 LNCS:648–677, 2021. ISSN 16113349. doi: 10.1007/978-3-030-77870-5_23/TABLES/2. URL https://link.springer.com/chapter/10.1007/978-3-030-77870-5_23.

[60] B. Liang and Z. J. Haas. Predictive distance-based mobility management for multi-dimensional pcs networks. *IEEE/ACM Transactions on Networking*, 11:718–732, 10 2003. ISSN 10636692. doi: 10.1109/TNET.2003.815301.

[61] C. D. Lima, D. Belot, R. Berkvens, A. Bourdoux, D. Dardari, M. Guillaud, M. Isomursu, E. S. Lohan, Y. Miao, A. N. Barreto, M. R. K. Aziz, J. Saloranta, T. Sanguanpuak, H. Sarieddeen, G. Seco-Granados, J. Suutala, T. Svensson, M. Valkama, B. V. Liempd, and H. Wymeersch. Convergent communication, sensing and localiza-

tion in 6g systems: An overview of technologies, opportunities and challenges. *IEEE Access*, 9:26902–26925, 2021. ISSN 21693536. doi: 10.1109/ACCESS.2021.3053486.

[62] H. H. H. Mahmoud, A. A. Amer, and T. Ismail. 6g: A comprehensive survey on technologies, applications, challenges, and research problems. *Transactions on Emerging Telecommunications Technologies*, 32, 2 2021. ISSN 21613915. doi: 10.1002/ETT.4233. URL https://dl.acm.org/doi/10.1002/ett.4233.

[63] M. Maouche, S. Ben Mokhtar, and S. Bouchenak. AP-attack: A novel user re-identification attack on mobility datasets. 2017. URL https://hal.archives-ouvertes.fr/hal-01785155.

[64] E. Y. Menta, N. Malm, R. Jantti, K. Ruttik, M. Costa, and K. Leppanen. On the performance of aoa-based localization in 5g ultra-dense networks. *IEEE Access*, 7: 33870–33880, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2903633.

[65] F. Mogyorósi, P. Revisnyei, A. Pašić, Z. Papp, I. Törös, P. Varga, and A. Pašić. Positioning in 5g and 6g networks - a survey. *Sensors 2022, Vol. 22, Page 4757*, 22: 4757, 6 2022. ISSN 1424-8220. doi: 10.3390/S22134757. URL https://www.mdpi.com/1424-8220/22/13/4757/htmhttps://www.mdpi.com/1424-8220/22/13/4757.

[66] C. Mouchet, J.-P. Bossuat, J. Troncoso-Pastoriza, and J.-P. Hubaux. Lattigo: a multiparty homomorphic encryption library in go.

[67] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1592:223–238, 1999. ISSN 16113349. doi: 10.1007/3-540-48910-X_16. URL https://link.springer.com/chapter/10.1007/3-540-48910-X_16.

[68] A. Panisson. pymobility v0.1 - python implementation of mobility models, May 2014. URL https://doi.org/10.5281/zenodo.9873.

[69] A. Paverd, A. Martin, and I. Brown. Modelling and automatically analysing privacy properties for honest-but-curious adversaries. 2014. URL https://www.cs.ox.ac.uk/people/andrew.paverd/casper/.

[70] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. 2009.

[71] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. 1977.

[72] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. 1978.

[73] M. Saily, O. N. Yilmaz, D. S. Michalopoulos, E. Perez, R. Keating, and J. Schaepperle. Positioning technology trends and solutions toward 6g. *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, 2021-September, 9 2021. doi: 10.1109/PIMRC50174.2021.9569341.

[74] SEAL. Microsoft SEAL (release 4.0). `https://github.com/Microsoft/SEAL`, Mar. 2022. Microsoft Research, Redmond, WA.

[75] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 11 1979. ISSN 15577317. doi: 10.1145/359168.359176. URL `https://dl.acm.org/doi/abs/10.1145/359168.359176`.

[76] B. Sun, B. Tan, W. Wang, M. Valkama, C. Morlaas, and E.-S. Lohan. 5g positioning based on the wideband electromagnetic vector antenna. 2021. URL `http://ceur-ws.org`.

[77] D. K. P. Tan, J. He, Y. Li, A. Bayesteh, Y. Chen, P. Zhu, and W. Tong. Integrated sensing and communication in 6g: Motivations, use cases, requirements, challenges and future directions. *2021 1st IEEE International Online Symposium on Joint Communications and Sensing, JC and S 2021*, 2 2021. doi: 10.1109/JCS52304.2021.9376324.

[78] A. Trivedi, C. Zakaria, R. Balan, and P. Shenoy. Wifitrace: Network-based contact tracing for infectious diseases using passive wifi sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5, 5 2020. doi: 10.1145/3448084. URL `http://arxiv.org/abs/2005.12045http://dx.doi.org/10.1145/3448084`.

[79] C. Troncoso et al. DP3T - decentralized privacy-preserving proximity tracing, 2020. URL `https://github.com/DP-3T/documents`.

[80] T. Wild, V. Braun, and H. Viswanathan. Joint design of communication and sensing for beyond 5g and 6g systems. *IEEE Access*, 9:30845–30857, 2021. ISSN 21693536. doi: 10.1109/ACCESS.2021.3059488.

[81] H. Yang, W. D. Zhong, C. Chen, and A. Alphones. Integration of visible light communication and positioning within 5g networks for internet of things. *IEEE Network*, 34:134–140, 9 2020. ISSN 1558156X. doi: 10.1109/MNET.011.1900567.

[82] Y. Zhang, J. Jin, C. Liu, and P. Jia. Indoor 3d dynamic reconstruction fingerprint

matching algorithm in 5g ultra-dense network. *KSII Transactions on Internet and Information Systems*, 15:343–364, 1 2021. ISSN 22881468. doi: 10.3837/TIIS.2021.01.019.

List of Figures

List of Tables