






A Privacy-Preserving Contact Tracing System based on a Publish-Subscribe Model

Mikaella F. da Silva  [Universidade Federal do Espírito Santo | mikaellaferreira0@gmail.com]

Bruno P. Santos  [Universidade Federal da Bahia | bruno.ps@ufba.br]

Paulo H. L. Rettore  [Franhouffer | paulo.lopes.rettore@fkie.fraunhofer.de]

Vinícius F. S. Mota   [Universidade Federal do Espírito Santo | vinicius.mota@inf.ufes.br]

 Department of Computer Science, Universidade Federal do Espírito Santo, Av. Fernando Ferrari, 514, Goiabeiras, Vitória, ES, 29075-910, Brazil.

Received: 23 October 2024 • **Accepted:** 21 May 2024 • **Published:** 11 August 2024

Abstract In the context of the COVID-19 pandemic, using contact-tracking apps and measures such as social isolation and mask-wearing has emerged as an efficient strategy to mitigate the spread of the virus. Nonetheless, these apps have raised privacy concerns. This paper introduces a technique for enhancing Privacy in contact-tracing systems while preserving the data for research purposes. The contact-tracing system employs a unique identifier signed with a key associated with the application and the user. In this system, mobile devices serve as sensors sending beacons, actively detecting nearby devices, and transmitting the identifiers of surrounding contacts to a cloud-based platform. When a user reports a positive COVID-19 diagnosis, a dedicated web service identifies and tracks the identifiers associated with at-risk contacts. The system uses a topic-based publish-subscribe broker, and each identifier represents an individual topic to abstract contact communication and disseminate alert messages. To assess the system's efficacy, we conducted a use case with twenty volunteers using the mobile application for two weeks, representing a small university campus. The quantitative results of the use case demonstrated the system's capability of analyzing potential virus transmission and observing user's social interactions while maintaining their anonymity.

Keywords: Contact tracing, Privacy, Anonymity

1 Introduction

The recent pandemic, caused by the novel coronavirus, has underscored the need for efficient mechanisms to control the spread of highly contagious diseases [Elavarasan and Pugazhendhi, 2020]. Besides isolation, social distancing, and mask-wearing, governments and other entities seek technological resources to assist in pandemic control. In this context, contact-tracing systems have gained significance ([Juneau *et al.*, 2023]). Indeed, these systems have proven helpful in controlling the spread of COVID-19 and were used in several countries ([World Health Organization, 2020]). The coronavirus is not the first pandemic humans have faced and may not be the last. Therefore, systems that help break the transmission chains are relevant for controlling pandemics and alleviating the health system. For individuals to feel comfortable using and sharing information with systems, their data privacy must be assured. However, people's willingness to engage with contact-tracing systems may fall short of expectations due to elevated data exposure and privacy concerns.

Typically, contact-tracing systems gather sensitive information, including names, social security IDs, email addresses, and {on, off}-line locations. Additionally, organizations recording individuals' contacts and tracing their location can create worry, as it may feel like constant surveillance. There is also the concern that these apps could be repurposed for user tracking and data misuse post-pandemic,

as discussed by Ahmed *et al.* [2020]. Recently, the consequences of massive data collection by Korean authorities were brought to the National Human Rights Commission, which classified them as a violation of human rights ([National Human Rights Commission of Korea, 2020]).

Contact-tracing systems can be grouped based on how they define closer contacts, such as i) *Bluetooth-based*, which can detect and record contacts using Bluetooth signal proximity. Users who tested positive can voluntarily report their status within the app, and other users in close contact with them are notified of potential exposure; ii) *GPS-based*, which tracks users' location history to identify possible visited places with higher exposure for virus transmission, allowing the creation of safe and unsafe map zones; and iii) *QR code-based*, which requires users to scan QR codes displayed at places they visit. If there is a COVID-19 outbreak in a specific location, these apps can quickly identify and notify individuals who checked in at that location. Contact can be defined as any individual detected by Beacon Bluetooth or GPS in the area of a user device. We argue that contact-tracing systems should not only manage users' contacts and proximity but also provide researchers with data for study-



Figure 1. Contact-tracing System Scenario.

ing these contacts. This, in turn, aids in understanding the dynamics of virus spread.

1.1 Problem definition

Before delving into the problem we address, it is important to note that the notion of privacy referenced in this text, as per Stutzman and Hartzog [2012], leans towards obscurity. In essence, while information may not be entirely private, it tends to be sufficiently obscure to afford reasonable privacy to individuals. The authors defined four factors of obscurity: i) *Search visibility*, which is how easy it is to find the user in search systems; ii) *Unprotected access to individual data*, which represents the degree of restriction to the data; iii) *Identification of an individual by direct or indirect exposure*, which concerns on disclosure an individual directly or indirectly; and iv) *Clarity*, which represents how an observer could comprehend a piece of information (data of the individual).

The contact-tracing system involves two main actors: device users and the overseeing authorities. We argue that to ensure privacy, the system must guarantee that neither actor can search for specific individuals and their personal data, which could potentially identify individuals in case of data leaks, compromising the system. However, it is necessary to use obfuscated identification to analyze contact dynamics, which is crucial for virus spread control. Furthermore, the system must offer a transparent, open-source approach that ensures reproducibility and adaptability for community use. Therefore, this study seeks to tackle the following research question:

Problem: *How to design an open-source contact-tracing system that safeguards user and authority privacy?*

1.2 Proposed solution

This work presents an anonymous contact-tracing system designed to protect users' privacy while maintaining insightful information for contact characterization and virus-spreading analysis. The system uses only the device's unique identifier anonymized with temporal public keys to identify contacts among users, ensuring privacy among users and authorities. Despite the emphasis on privacy, the system allows network contact and virus spread analyses. It comprises three main components: a user-friendly mobile application, a Message Queuing Telemetry Transport (MQTT) broker for efficient message sending over the Internet, and web services responsible for storing, managing contacts, and issuing alerts in case of user infection. The data containing contacts and alerts issued can be modeled as a social graph, allowing researchers to analyze the dynamic of the virus spreading with full respect to privacy.

Figure 1 illustrates the proposed system through a possible risk scenario, such as schools and universities, where students and teachers share enclosed spaces (A). Users' devices periodically broadcast an anonymous identifier, locally store the received broadcasts, and, at regular intervals, send them to the server. If users test positive for COVID-19 (B), they can notify the system. A cloud-based service filters users

who had contact with the notifying user, categorizes risk groups based on the duration and proximity of contacts, and sends alerts to them (C). Later, researchers can use the contact database to study the dynamics of virus spread (D) and implement/improve measures to contain the spread. Moreover, it is impossible to infer which user reported the disease or who received the notification. In summary, the contributions of this study are as follows:

- We designed, developed, deployed, and assessed a contact-tracing system that ensures user privacy from other users and the system administrator's perspective. That means privacy is ensured on both sides.
- We designed a weighted graph approach, considering the distances between users and the duration of contact to analyze the contacts and the impact on virus transmission.
- Through a real use case scenario with 20 users over two weeks on a university campus, we conducted a quantitative analysis showing the system's reliability.
- The contact-tracing system is released as an open source with the dataset containing 866 contacts recorded, of which 341 contacts exceeded 15 minutes.

1.3 Paper organization

The rest of this paper is organized as follows: Section 2 discusses the background and related work of contact-tracing systems. The proposal is described in Section 3. Section 4 presents the use case, its quantitative data analysis, and the dynamics of the contact graphs of the system on campus. Section 5 discusses the limitations and challenges privacy-based contact-tracing systems face. Finally, Section 6 presents the final considerations.

2 Background

Contact-tracing systems have been used to identify, assess, and manage individuals exposed to a disease to prevent further transmission. The concept is to locate individuals who have had contact with an infected person and provide recommendations, such as testing and social isolation. World Health Organization [2020] provides guidance on defining a *contact risk* in the context of COVID-19. Risk contacts represent individuals within 1 meter of a COVID-19 case for more than 15 minutes, with the onset of the disease confirmed between 2 and 14 days ago.

There are three commonly used architectures in contact-tracing systems: centralized, decentralized, and hybrid ([Ahmed et al., 2020; Morio et al., 2023]). In the centralized architecture, users must register with the server, providing information that can be used to identify and contact them if necessary, and the server generates temporary identifiers (IDs) for each device. In the decentralized architecture, users are responsible for creating temporary IDs and identifying if they have had contact with an infected person. The Private Automated Contact Tracing (PACT) protocol defines that devices are responsible for producing a pseudonym, shared with other users in proximity ([Rivest et al., 2020]).

The hybrid architecture suggests that the devices should generate and manage temporary IDs to shield their privacy and anonymity. At the same time, the centralized server should take on the roles of risk analysis and user notification.

According to Cho *et al.* [2020], there are three desirable concepts of privacy in the context of contact-tracing systems: i) *privacy from snoopers*, someone sniffing the network to obtain the tokens; ii) *privacy from contacts*, which means a user is unable to define who declared as infected, based only in the app information; and iii) *privacy from authorities*, where the authorities responsible for the system also are unable to disclose someone who declared positive diagnosis. The next section presents the related works on the contact-tracing systems grouped by the three types of architectures (centralized, decentralized, and hybrid), the capability of safeguarding the user and authority privacy, the enabled technology employed, and the dissemination that covers the code and the data acquired.

2.1 Related work

Contact-tracing systems that use centralized architecture are widely found in the literature. TraceTogether is an example of a digital centralized Bluetooth-based contact-tracing system developed and used in Singapore to combat the spread of COVID-19 ([Stevens and Haines, 2020]). McLachlan *et al.* [2020] propose a centralized system that combines Bayesian infection prediction and traditional tracing to minimize pandemic issues. [Danquah *et al.*, 2019] present a proof-of-concept study of the Ebola issue using a centralized contact-tracing mobile app in Sierra Leone, highlighting that the contact-tracing systems may help in different public health crises.

Castelluccia *et al.* [2020] introduced the ROBust and privacy-preserving proximity Tracing (ROBERT) scheme that implements a centralized architecture. ROBERT uses a federated server system and temporary anonymous identifiers to provide robust security and privacy assurances. ROBERT is a potential candidate for the Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT). BlueTrace, proposed by Bay *et al.* [2020], is another centralized contact-tracing system. It implements a privacy-preserving protocol designed for cross-border, community-driven contact tracing, facilitating effective contact tracing while maintaining user privacy and data security.

Ahmed *et al.* [2020] present the differences between the BlueTrace and ROBERT. While BlueTrace stores personal information (phone numbers), ROBERT stores only anonymous identifiers referred to as “EphIDs”, providing a higher level of privacy. The protocols also differ in the notification process. ROBERT requires all users to frequently check their EphIDs used with the server to determine if they are flagged as at risk. In contrast, with BlueTrace, health authorities can proactively notify at-risk users. The discussion of how a centralized and GPS-Based contact-tracing mobile application handled COVID-19 in India was proposed by Gupta *et al.* [2020]. The authors showed that the app called *Aarogya Setu*, applies different concepts of data science, despite the authorities and analysts having full access to users’ sensitive data.

Recently, Wahid *et al.* [2023] proposed the COVIC-T, a centralized system capable of gathering user data, detecting and monitoring it with semi-automated and improved contact tracing. The main idea concerns monitoring symptoms in real-time to predict contaminated areas and imposing a smart lockdown on these areas. Smith *et al.* [2024] propose a WiFi proximity inference technique to calculate contacts and perform tracing. This centralized architecture requires several personal information. Therefore, user privacy regarding the administrators of the system is questionable.

Moving into the decentralized approaches, these solutions appear less frequently than centralized in the literature, as shown by Vaudenay [2020]. Troncoso *et al.* [2020], for example, propose the Decentralized Privacy-Preserving Proximity Tracing (DP3T), which claims to offer a decentralized and privacy-preserving solution. Canetti *et al.* [2020] propose a Bluetooth-based scheme for providing fine-grained and timely alerts to users near an infected individual. Another decentralized proposal is Private Automated Contact Tracing (PACT), which uses smartphones for contact tracing based on Bluetooth proximity ([Rivest *et al.*, 2020]). Recently, the authors Fahliani *et al.* [2023] proposed of a decentralized privacy-preserving contact tracing protocol, called DPACT, where users are active or passive ones, the first send anonymized BT-beacons, the second only listen to beacons. Liu *et al.* [2023] proposed a decentralized technique that combines non-interactive zero-knowledge proof and multi-signature with public key aggregation offered by blockchain. Another blockchain approach is the Blockchain-Driven Contact Tracing (BDCT) proposed in Li *et al.* [2024], authors use blockchain, RSA encryption, and reputation system to secure users’ identity/privacy.

Apple-Google Apple and Google [2023] announced a partnership releasing a service of privacy-preserving contact tracing. The paper by Gvili [2020] assesses the Apple-Google solution and identifies vulnerabilities. Also, it is unclear whether the Apple-Google approach is a centralized, decentralized, or hybrid system due to the proprietary implementation.

Based on our review, not many hybrid contact-tracing approaches are available nowadays. Epione ([Trieu *et al.*, 2020]) is a contact tracing App aiming to alert users about exposure to COVID-19 in time while protecting the privacy of users’ contacts from central services and other users. Lee *et al.* [2021] draw comparisons among centralized and decentralized approaches to contact-tracing protocols and propose a hybrid protocol. In this protocol, each user locally generates a key and sends it to a central authority. The central authority generates a new key using the user’s and authority’s keys and sends it back to the user. This is called the “re-key”. Users compute tokens using their key and the re-key and, locally, maintain a list of received tokens when users come into contact. If a user tests positive, they send the list of received tokens to the server, which publishes it publicly. Users who receive the list locally, derive their tokens and check if they are in the received list. If a token matches, the user has been in contact with an infected person.

The system outlined in this study alleviates the user’s burden of self-identifying potential risks, in contrast to the approach presented in prior studies by Lee *et al.* [2021] and

Table 1. Related Work

System (ref.)	Objective	Architecture	Privacy Contacts	Privacy Authorities	Open-source	Enabled Technology	Data Available
TraceTogether Stevens and Haines [2020]	To combat COVID-19 in Singapore	Centralized	✓	✗	✓	BLE-Based	✗
BayesCOVID Surveillance McLachlan et al. [2020]	Improve contact tracing privacy using only Prob. user has Covid-19, GPS-location, Age-group information	Centralized	✓	✗	✗	Bayesian network	✗
ECT app Danquah et al. [2019]	Track contacts of Ebola cases	Centralized	✗	✗	✓	BLE-Based	✓
ROBERT Castelluccia et al. [2020]	Centralized bluetooth-based system for the joint contribution in the PEPP-PT ¹	Centralized	✓	✗	✗	BLE-Based	✗
BlueTrace Bay et al. [2020]	It aims to facilitate effective contact tracing while maintaining user privacy and data security	Centralized	✓	✗	✓	BLE-Based	✓
COVIC-T Wahid et al. [2023]	COVID-19 detection through IoT-based semi-automated and improved contact tracing	Centralized	✓	✗	✗	IoT-Based	✓*
Aarogya Setu The contact tracing Gov [2020]	Identifies Potential Covid-19 cases using online self-assessment and traces contacts using location data	Centralized	✗	✗	✗	GPS	✗
Smith's framework Smith et al. [2024]	To estimating contact between users using Wi-Fi	Centralized	✓	✗	✗	WiFi-based	✗
DP3T Troncoso et al. [2020]	Propose a newly decentralized solution for contact tracing	Decentralized	✓	✗	✗	BLE-Based	✗
Canetti-Trachtenberg-Varia Canetti et al. [2020]	Provide alerts to users who have been near to viral infected individual	Decentralized	✓	✗	✗	BLE-Based	✗
PACT Rivest et al. [2020]	Alternative decentralized protocol to contact tracing	Decentralized	✓	✗	✗	BLE-Based	✗
Momeng's framework Liu et al. [2023]	To protect the privacy of close contacts of confirmed patients.	Decentralized	✓	✗	✗	BLE-Based and others.	✗
DP-ACT Fahlani et al. [2023]	DP-ACT allows active and passive participants. Active participants broadcast BLE beacons with pseudo-random IDs. Passive just listen to the broadcasted BLE beacons.	Decentralized	✓	✗	✗	BLE-Based	✓*
BDCT Li et al. [2024]	To propose BDCT framework Blockchain and RSA based including a reputation system.	Decentralized	✓	✗	✗	BLE-Based	✗
Apple-Google Apple and Google [2023] Michael and Abbas [2020]	A partnership to provide a service of privacy-preserving contact tracing	-	-	✗	✗	BLE-Based	✗
Epione Trieu et al. [2020]	A Contact Tracing app Privacy aware that Alerts user about exposure in the last 21 days	Hybrid	✓	✓	✗	BLE-Based	✗
Lee-Hybrid-Protocol Lee et al. [2021]	Provide a hybrid approach while provide privacy to users	Hybrid	✓	✓	✗	-	✗
Proposed Solution	Provide a hybrid contact tracing system while keep privacy of users to users and authorities	Hybrid	✓	✓	✓	BLE-Based	✓

*The work uses a third-party dataset with specific information about COVID-19 or face-to-face interactions without geo-location information.

Trieu et al. [2020]. This task entails significant computational resources and the potential exposure of user-sensitive contact data due to users needing to download tokens from the infected person's device. Additionally, the system does not collect personal information, as the BlueTrace or government solutions (e.g., TraceTogether) protocols do, aiming for anonymity. Still, as presented in the ROBERT, DP3T, and Epione protocols, it manages to notify users without requiring them to constantly check if they are at risk.

More about the contact tracing approaches is available in Vaudenay [2020]. The authors discussed the contact tracing dilemma among centralized or decentralized implementation, as well as their security and privacy issues. Also, comprehensive reviews of mobile computing to tackle the COVID-19 pandemic through contact tracing are presented by Ali and Khan [2023]; Jiang et al. [2022]; Duan and Deng [2022]. The authors have classified solutions based on different features or functions related to COVID-19 or the system approach, such as tracking suspected cases, catching symptoms, surveillance, dashboard and analysis, contact tracing, system security, and privacy, etc. We also, highlight the novelty and scientific motivations of the work based on recent literature. The reader can also get relevant and related systematic reviews in Leung et al. [2024]; Rizi et al. [2024].

Table 1 presents a non-exhaustive list of related works, providing a feature comparison. Each row shows the reference, the system/app's main objective, the architecture employed in the proposed contact-tracing system, and pri-

vacancy concerning the contacts (e.g., anonymity between users) and authorities (e.g., service controllers cannot track specific users). Followed by the enabling technology employed and whether the system is freely accessible to the community and the availability of the resulted dataset. In summary, we have introduced a hybrid architecture for contact tracing that leverages BLE technology.

Differently from the current literature, this study brings privacy among users, meaning that users cannot identify who informed a positive test only with the app information. Thus, the main contribution is providing privacy for contact users and authorities which cannot identify the users individually. Besides user-to-user and authority-to-user privacy, the framework designed is open-source and it is available to the scientific community, see Section 6.

3 The contact-tracing system

Contact-tracing systems identify, assess, and manage individuals exposed to a disease to prevent further transmission [World Health Organization, 2020]. The objective is to locate individuals who have had contact with an infected person, i.e., individuals with the potential to be infected, and provide recommendations such as testing and social isolation. Currently, most people carry devices capable of contact tracing through Bluetooth Low Energy (BLE) or Location-based services. In this study, we define BLE as the technology

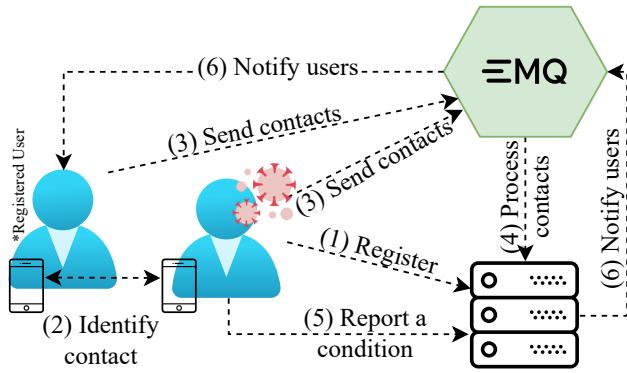


Figure 2. Contact-Tracing Overview.

for contact tracing due to its lower energy consumption and, most importantly, its avoidance of storing sensitive information such as users’ geolocation.

The system architecture includes a mobile application, cloud-based web services, and an MQTT broker. The mobile application was developed in Java for the Android operating system. At the same time, the web services were implemented in the Go programming language, utilizing a relational PostgreSQL database and Redis as a cache. The mobile application is responsible for identifying nearby individuals, while the web services track an infected user’s contacts to detect potentially at-risk individuals and notify them. MQTT enables the mobile device to publish only the contacts that have occurred, and each mobile device subscribes to its topic, corresponding to its unique identifier. In this way, in the event of notifications, the public notification service sends messages to the topics of those who should be alerted, and the messages are delivered asynchronously via an MQTT broker. In this work, we use EMQ², an open-source MQTT broker for IoT, IIoT, and connected vehicles. It supports MQTT broker clustering, ensuring the system’s scalability.

Figure 2 provides an overview of the architecture and interactions among the services of the proposed system, as summarized below:

- (1) **Register:** Initially, the user needs to register in the system. In this case, a system identifier generates a Universal Unique Identifier (UUID) for the user.
- (2) **Identify contact:** The application on the device uses this identifier to identify nearby contacts. The device periodically sends beacons with its identifier using BLE.
- (3) **Send contacts:** Upon receiving a beacon, it is possible to estimate the distance between these devices based on the received signal strength. The devices send the dataset containing the contacts at predefined intervals via the MQTT protocol.
- (4) **Process contacts:** The service validates the authenticity and integrity of the message and saves this contact in a database or discards it if it is invalid. One of the users in contact may have been recently diagnosed with COVID-19. Therefore, it is important to verify this information for each received contact.
- (5) **Report a condition:** Upon receiving a positive diagnosis, the user must report it to the system. The user informs the diagnosis date and sends it to the web server. Asynchronously, a tracking web service will retrieve

from the database all identifiers that have been in contact with this user in the last 15 days, with contact lasting more than 15 minutes and a distance of less than 2 meters between them, following WHO recommendations.

- (6) **Notify users:** This service will notify all users who fit the risk contact rule. This notification is sent via a message published in a specific MQTT server topic for the at-risk user. Even if the user is offline, the device will receive an alert when it connects to the Internet.

It is important to emphasize that the validation of the diagnosis would require personal information, going against the goals of the proposed system. Consequently, it may be susceptible to false positives. The following subsections provide a detailed overview of the services related to user registration in the system, the detection of nearby individuals, the processing of received user contacts, and the system’s implementation.

3.1 Anonymizing and registering a user

To perform contact tracing, users must install a mobile application on their devices. The system requires that the device use a trustful Network Time Protocol (NTP). To ensure anonymity, the application generates the registration data, which includes a pair of asymmetric keys and the system identifier. Mobile device operating systems provide a unique system identifier. For this work, we utilized the Service Set Android Identifier (SSAID), which represents the sole information associated with the user’s device. The SSAID is a 64-bit number expressed as a hexadecimal string, unique for each application signature key, user, and device combination. The Android system mandates that all applications be digitally signed with a certificate before installation or updates. Therefore, the identifier is unique to the application obtaining it, even if it is uninstalled and reinstalled. Consequently, if a user changes or resets their device, a new system ID is acquired, necessitating re-registration in the contact-tracing system. It is essential to highlight that hardware identifiers, such as MAC addresses, can be easily duplicated and should thus be avoided.

After installation on the device, the mobile application generates a pair of keys using the elliptic curve algorithm and stores them in the application’s memory. The public key is shared with the server during registration, while the device uses the private key to sign messages. The server can verify the message by verifying the signature with the user’s stored public key, ensuring message integrity and access control.

Figure 3 illustrates the user registration flow within the system. The server generates a pair of private and public keys, periodically. The mobile application first searches for its ID, if not found and the user clicks on register screen, it creates its own private and public key pair. The server registration service receives a pair of keys and an identifier generated by the server based on the user’s SSAID. Situations in which the application will not find this information include the user’s first access to the application and after clearing the application data. When the pair of keys and/or the identifier are not found, the application prompts the user for registration. Subsequently, the application generates the asymmetric key pair

²<https://www.emqx.com/en>

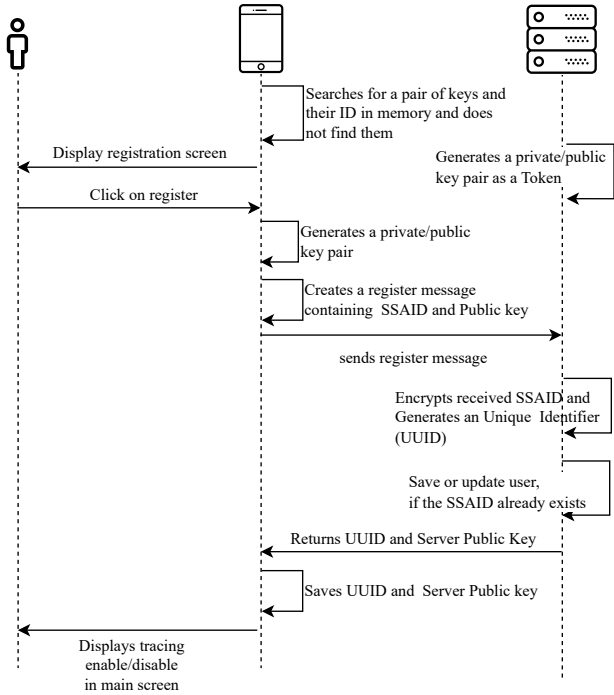


Figure 3. Process of registering a user in the system.

and stores it in memory. The registration message requires the SSAID (device ID) and the public key (pk), both represented in hexadecimal format, as exemplified in Listing 1.

Listing 1: Device register message

```
{
  "pk": "3059301306072a8648...",
  "deviceId": "65ab8c72d904..."
}
```

The application sends the public key and SSAID to the server through a remote procedure call (gRPC). SSL/TLS over gRPC can guarantee authentication and encryption of data exchanged between the cloud and devices. Subsequently, the server encrypts the SSAID and generates a 16-byte UUID, which the application will utilize for beacon advertising. The SSAID cannot be shared with other users, as it is confidential information used for user registration or updates. The UUID encrypted SSAID, and public key are stored in the database. If a record with the same encrypted SSAID already exists, the public key is updated, indicating that the user remains with the same device but has uninstalled the application. Since the SSAID is unique for that application and user on that device, it can be assumed to be the same individual. In this case, the generated UUID is discarded, and the identifier in the database is retained. The server returns the UUID to the device, which will be persisted in the application’s memory.

The server must regenerate its private and public key, periodically, to avoid a malicious user from capturing UUID and impersonating another user. The server announces its public key, and mobile clients encrypt their UUID with this public key before announcing it. Therefore, spurious and malicious users can attempt to impersonate another user at the maximum time of the duration of the server public key. The next section details how server controls their private and public keys.

3.2 Identifying and announcing contacts

This work assumes the devices possess a wireless communication interface using the BLE protocol. BLE has been designed to be a low-power wireless technology, suitable for applications that prioritize energy efficiency and do not require high transfer rates ([Gupta, 2016]). Furthermore, BLE allows devices to function as beacons. A beacon is a concept of transmitting small pieces of information, also called beacon advertisements.

Figure 4 illustrates the flow of the contact identification service between devices and the transmission of contacts to the cloud. To enable contact-tracing, each mobile device initiates the *Beacon Announcement Service* in the foreground for broadcasting its identifier and listening for new beacons. After receiving a new beacon, it decodes the beacon and saves it as a contact. Periodically, the mobile device queries the last contacts (1), creates a message with the list of contacts and signed (2), and finally, publishes a MQTT message (3), which will be received by the server. Next, we detail the identification and announcement of the list of contacts.

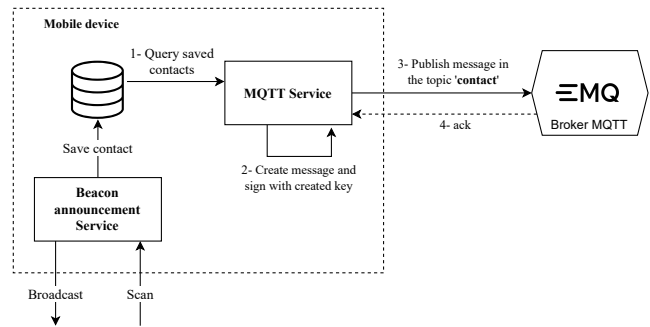


Figure 4. Detect nearby devices and send the contact to the web service.

3.2.1 Identifying contacts

To identify mobile devices, they must broadcast beacons. Each mobile device executing the *Beacon Announcement Service* broadcasts its beacons using the *AltBeacon* protocol ([AltBeacon.org, 2014]). A message in *AltBeacon* format is encapsulated as a payload within the data structure for BLE advertisements. It consists of a 1-byte length field, a 1-byte type field, and a 2-byte company identifier. The additional 24 bytes contain the advertisement data, a 20-byte beacon ID, and a 1-byte Received Signal Strength Indication (RSSI) reference. Mobile devices act as beacons by periodically encrypting their UUID with the server public key, and inserting and transmitting this encrypted UUID in the Beacon ID field. Through RSSI, it is possible to estimate the distance between devices³.

The device advertises beacons every 15 seconds. We established beacon announcements every 15 seconds after the prior exploration test. Extending this interval may result in losing contact, while shorter intervals offer a detailed snapshot of the contact graph at any given moment. However, given the relatively static scenario, students in a class, we found that intervals shorter than 15 seconds neither enhance

³The Android Beacon Library API provides a method for estimating the distance between objects based on RSSI.

the final contact graph nor justify the increased energy consumption. Upon receiving a beacon, the device decodes and extracts the encrypted *UUID* from the *Altbeacon* protocol to generate contact information and store it in a local database. A contact is defined as a tuple containing the received identifier, the timestamp of contact start, the timestamp of contact end, the approximate average distance, the received signal strength, and the user's battery level when receiving that beacon. Suppose the user receives beacons with the same encrypted *UUID* within a 2-minute interval with a distance variation of no more than 1 meter. In that case, the last saved contact record in the database for that identifier is updated with a new contact and timestamp, and the average distance is recalculated. A new contact record is created if the time between beacon messages exceeds 2 minutes or the distance variation is greater than 1 meter.

3.2.2 Announcing the list of contacts

The mobile application announces a list of contacts at intervals of 30 minutes. This interval is based on the definition of contact risk. By the OMS definition, a user is at risk in case of contact greater than 15 minutes closer to 2 meters. Therefore, after 30 minutes, if a user has a contact of risk, it is supposed that the local database contains all information about this contact. Furthermore, contact messages must be published asynchronously for anonymous issues. In such a way, in case of receiving a notification of the risk of COVID-19, a user cannot identify exactly who he/she encountered previously.

Each announcement consists of a list containing 60 Contact records from the local database, not transmitted, based on the observations of Hossmann *et al.* [2011], which characterized mobility of users based on several Wi-Fi traces as a graph. The authors observed that most of the community (clusters) of users are formed by less than 10 nodes, with some achieving 50 nodes. As explained in Section 3.2.1, the tuples of the same devices encountered within less than 2 minutes are aggregated in the mobile application, which occurs in high frequency, assuming that the users' social network is stable over time. Therefore, the list of tuples of contacts per upload is expected to be smaller than 20. However, since users may have encountered many unknown users once, we defined the list of contacts as the last 60 records. We highlight that if a user presents more than 60 contacts in an interval of 30 minutes, the remaining contacts will be uploaded to the next round.

The authenticity and integrity of each contact record are ensured through a signature created using the user's private key. This information is formatted into a JSON string and signed with the private key stored on the user's device. Consequently, the signature field is added to the message to validate it with the cloud-based contact receiving service. The code snippet in the Listing 2 illustrates a contact record ready to be sent. To submit to the cloud, the device acts as a publisher connected to a private MQTT broker of the system and publishes messages to a topic called "contact". A cloud-based contact processing and storage service subscribes to this same topic and broker.

Listing 2: Contact message with signature

```
{
  "id": 1,
  "contact": {
    "otherUser": "123e4567-e89b-12...",
    "firstContactTimestamp": 2435465634,
    "lastContactTimestamp": 2435465815,
    "distance": 120, //cm
    "rssi": -15,
    "batteryLevel": 51
  },
  "user": "3d0ca315-aff9-4fc2-be61-...",
  "signature": [48,69,2,33,0,204,95,...]
}
```

The service must process each contact published by a user in the "contact" topic asynchronously. To achieve this, a service called *ContactsProcessing* acts as an MQTT client subscribes to this topic, registering a handler to process the received messages. Upon receiving a message, the registration service validates its authenticity and integrity, and checks if it contains all the expected fields. The server keeps a table with a timestamp in which the $n - last$ private-public keys were created and announced. The server uses the timestamp for each contact tuple with the encrypted *UUID* to select the correct private key, decrypting the *UUID*. The size of n depends on the frequency the server changes the private-public keys.

With all information checked and decrypted, a server service stores the contacts in an SQL database. Subsequently, for each new contact, the service forwards it to the COVID-19 risk contact tracer, and if necessary, the user will be asynchronously notified, as explained in the next section.

3.3 Tracking and notifying risk contacts of a positive diagnosis

To report a positive test result for COVID-19, a user must provide the date of symptom start and the date of receiving the positive diagnosis. In the case of asymptomatic individuals, it is recommended to report the date of symptom start as equal to the date of diagnosis. The positive diagnosis message contains the dates provided by the user and the date on which the user reported, i.e., the date they sent the message to the server. It is important to emphasize that, to ensure that no personal information is disclosed to the system administrator, the only validation performed is the message's signature. However, this leaves the system vulnerable to false positives.

Upon receiving the message, the service validates the message's signature with the user's public key. The diagnosis is stored in the database and cache if it is valid. It is essential to store this diagnosis in the cache with a 15-day expiration period. In such a case, the system can quickly retrieve infected users when a user comes into contact with an infected person. Therefore, it is important to notify users of the risk of contagion. The notification of users at risk of contagion requires two services operating asynchronously: the risk contact tracker and the risk alert notifier.

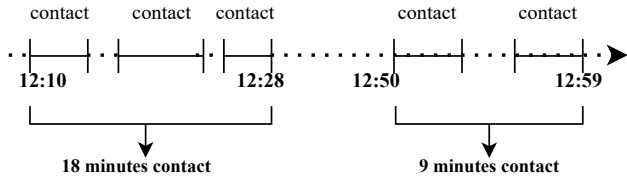


Figure 5. Aggregation of contacts between two same people.

3.3.1 Risk contact tracker

According to the WHO [World Health Organization, 2020], when in contact with an infected individual for more than 15 minutes at a distance of less than 2 meters, there is a high risk of COVID-19 transmission. Such contacts should be reported for testing and potential isolation. Otherwise, the risk is considered low. In this sense, contact-tracing is the service responsible for identifying all contacts of an individual who has reported a positive diagnosis and filtering those contacts with a risk of transmission, following WHO recommendations. Algorithm 1 describes the process of tracking high-risk contacts. Initially, the tracker executes a query in the database, filtering contacts sent by the source identifier within 15 days before the diagnosis date. Then, the total duration between contacts is calculated, i.e., the time between subsequent notifications among the same contacts.

Contact data is aggregated so that if the time difference between two contacts involving the same users (UUIDs) is less than 20 minutes, it is assumed to be the same contact. In other words, the time difference between these two contacts is included in the contact duration, and they are considered the same. Figure 5 illustrates an example of this procedure. From the notification of a positive diagnosis, the query returns a set of tuples $[UUID_p, UUID_c, dist, dur]$, representing the identifier of the user who reported positive, the identifier of a contact, the distance of this contact, and the duration of this contact tuple, respectively. In this manner, the contagion risk is calculated as follows in 1.

$$Risc(UUID_p, UUID_c) = \begin{cases} 1, & \text{if } (dist \leq 2) \wedge (dur \geq 15) \\ 0, & \text{else} \end{cases} \quad (1)$$

For cases where $Risc = 1$, the risk notification service will notify the user $UUID_c$.

3.3.2 Notifying risk contacts

The notifier is a service responsible for notifying users at risk of contagion. One of its functions is to send messages to a particular user's contagion risk notification topic. We utilize the MQTT topic model to notify users at risk of contagion to ensure the system's anonymity. After registering in the system, the device subscribes to a topic defined as a string containing its unique identifier ($UUID$). The notification service takes as input the tuples resulting from Algorithm 1, which contain risk contacts. For each risk contact ($UUID_c^i$), the notification service publishes a message to a topic following the format `notification/UUID` as shown in Listing 3.

Algorithm 1: Contact Risk Calculation

```

/* Data input as tuples of a query filtering
   contacts within 15 days with  $UUID_p$  */
Data: Tuples of contacts  $UUID_p$  in the last 15 days
Result: Tuples  $[UUID_p, UUID_c, dist, dur]$  where
            $Risc(UUID_p, UUID_c) > 1$ 
/* Aggregate the duration of contacts lower
   than 15 minutes. */
for each contact tuple
  ( $Timestamp_i, UUID_p^i, UUID_c^i, dur_i$ ) in the sorted
  contacts list do
  if next contact tuple
    ( $UUID_p^{i+1}, Contact_{i+1}, Timestamp_{i+1}$ ) exists
  then
     $T_{diff} = Timestamp_{i+1} - Timestamp_i$ 
    if  $T_{diff}$  is less than 15 minutes then
      Merge the contact tuples
      ( $UUID_p^i, UUID_c^i, Timestamp_i$ ) and
      ( $UUID_p^{i+1}, Contact_{i+1}, Timestamp_{i+1}$ )
      into a single contact with an updated
      duration;
    end
  end
end
Initialize an empty list  $ResultTuples$ 
for each contact tuple
  ( $UUID_p, UUID_c^i, Distance_i, Duration_i$ ) in the
  merged contacts list do
   $R(UUID_p, UUID_c^i) \triangleright$  calculated with Equation 1
  if  $R > 1$  then
    Add the tuple
     $[UUID_p, UUID_c^i, Distance_i, Duration_i]$  to
     $ResultTuples$ ;
  end
end
Return  $ResultTuples$ 

```

Listing 3: Contact of Risk message

```

{
  "risk": true,
  "message": "Contact of risk."
}

```

For example, a user with $UUID$ `f234454c-f6d4-a0fa-df2f-4911ba9ffa6` subscribes to the topic `notification/f234454c-f6d4-a0fa-df2f-4911ba9ffa6`. If they become a risk contact, the notification service will publish the following message to this same MQTT topic. A cache prevents database queries for every incoming task, which would take longer than a direct query. The system uses a cache to store risk contacts for 15 days to enhance query efficiency and prevent users from receiving duplicate notifications. After this period, another service removes expired notifications from the cache daily and notifies users that they are no longer at risk.

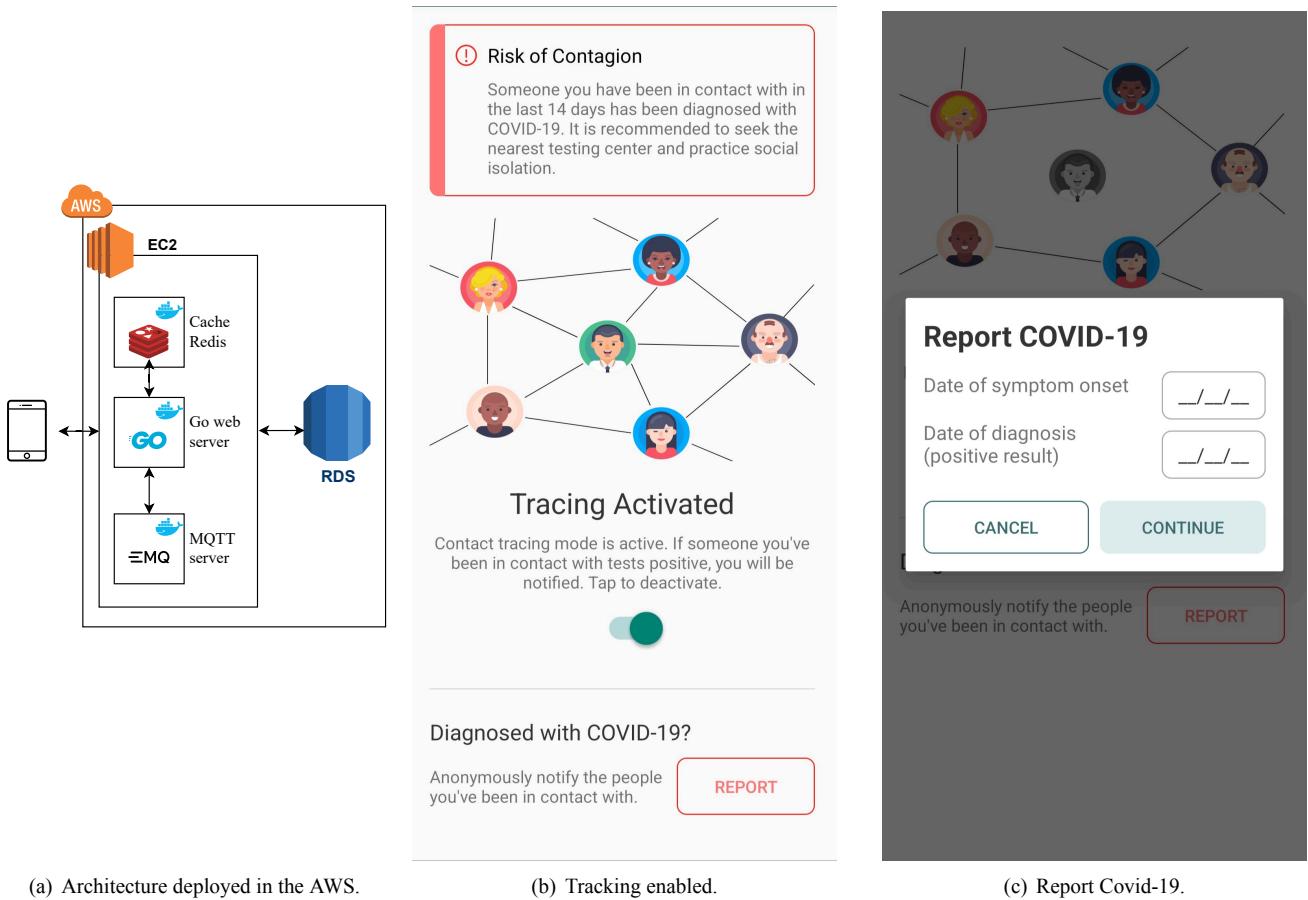


Figure 6. Deploy the web application and mobile tracking application screens.

3.4 Prototype: Development and deployment

4 Use case: Monitoring a campus

3.4.1 Cloud Services deployment

To enable the mobile application to communicate with the web server and MQTT broker, they must be accessible on the internet. The application deployment was executed on the Amazon Web Services (AWS) cloud computing platform. The resources utilized comprised Amazon RDS for the PostgreSQL database and an Amazon EC2 instance to run the servers. Due to financial concerns, only one EC2 instance was employed, though the ideal setup involves separate machines for the servers. Figure 6(a) illustrates the cloud-based architecture of the application. The web server, Redis cache, and MQTT broker were virtualized using Docker containers. The Docker image for the web server was created on Dockerhub to be utilized within the EC2 instance, providing easy updates.

3.4.2 Mobile app

The application was developed using Java, and an APK file was generated for user installation. Figure 6(b) illustrates the screens where the tracking feature is activated, and the user is notified of the risk of contagion. One can observe the “Report COVID-19” button on the tracking screen. Figure 6(c) depicts the information requested from the user upon clicking this button, which includes the date of symptom start and the date of receiving a positive diagnosis.

The Federal University of Espirito Santo (UFES) students in Brazil used the mobile app to assess and validate the system. The goal was to evaluate the non-mandatory adoption and the capability to perform risk contact data analysis without compromising user privacy. Initially, a widespread project promotion was conducted among university students. Afterward, a form available to volunteers details the system’s objectives and data collected while using the app. All volunteers had to consent to this information before installing the app. Moreover, the volunteers were instructed to use the system for a few hours a day, and the usage was recorded over the two weeks of the experiment.

Over these two weeks, the generated contact messages were sent to the cloud for processing. We highlight that the server created only one public key to encrypt *UUID*. Therefore, the encrypted *UUID* for every user contact was kept equal during this experiment. To simulate an infection, two volunteers were randomly selected to receive a positive test notification for COVID-19, and the contact graphs and their properties were examined using contact identification messages and contagion risk analysis. Note that because the positive diagnoses were artificially created, the notifications did not indicate genuine risk to users.

4.1 Data Visualization and Analysis

Anonymity is a pivotal contribution of the system. Despite the participants remaining anonymous, this system enables researchers to analyze the spread of contagion and contact patterns, primarily through graph networks. We construct these graphs with vertices symbolizing individuals and edges representing interactions or contacts between them. When an individual reports being infected, the system records this by marking the corresponding vertex in the graph as infected, potentially leading to further transmission through connecting edges. It's worth noting that a vertex with a higher number of edges is more susceptible to infection and is more likely to transmit the infection to other vertices.

Moreover, the edges in these graphs can carry additional information through weights. In the context of a contact graph, these weights may represent various parameters, such as the duration of the contact, the distance between individuals, and the frequency of interactions between the respective vertices. We examine the cumulative distribution of two critical factors for assessing the risk of COVID-19 transmission: the average duration and distance between contacts and the average degree of vertices in the contact graph. These cumulative distributions facilitate the computation of the probability that an individual within the sample has been exposed to the virus.

4.2 Experiments

4.2.1 Data summary

In total, 20 users registered in the system, and 866 contact data points were collected between November 6, 2022, and November 20, 2022. Two individuals were randomly selected to report positive diagnoses. Based on these reports, three notifications of contagion risk were generated. From 20 users, 17 generated contact data. The contact data was aggregated based on continuous contact, using Algorithm 1. Note that data will be aggregated when the interval between consecutive contact is less than 15 minutes. Table 2 provides a summary of the collected data. One can argue the statistical relevance of 20 users. However, it is important to note that this investigation occurred in a real-world setting as part of a specific use case on a campus. Observing authentic human behavior within this context was possible despite the sample size, which holds significant relevance for the study. Additionally, the main concern is to demonstrate the reliability of the methodology employed, laying a solid foundation for future expansion of the use case.

4.2.2 Quantitative Analysis

Figure 7 displays the number of contacts the server receives daily during the testing period. Notably, there were days

Table 2. Summary of data collected by the system.

System Data	
Registered users	20
Users with contacts	17
Contact records	866
Constant contacts	341

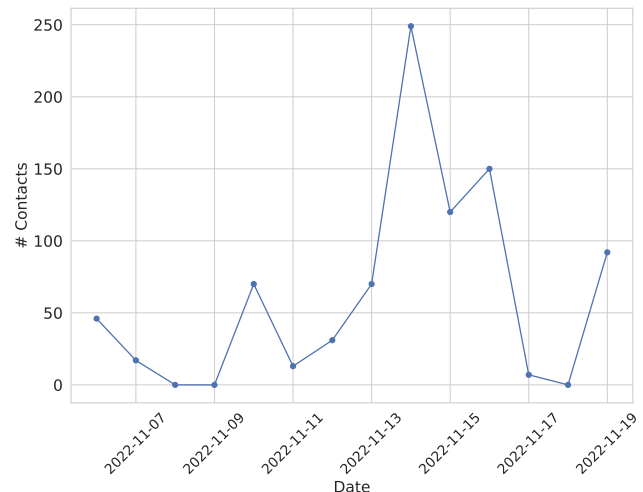


Figure 7. The number of contact records per day.

when no contacts were reported, indicating that users likely forgot to activate the system for some hours, as recommended. On November 13, 2022, the volunteers were invited to a testing event where they were explicitly instructed to use the app. Therefore, a significant increase in reported contacts was observed during this event. Notice that, as part of the scientific methodology adopted, the system was exposed to different scenarios, from ideal, where all the users activate the system daily, to worst scenarios, where most users forget to activate the system, leading to sparse data to be analyzed. People likely forget or intentionally deactivate the system, which was a challenge faced by the authorities during the pandemic. Therefore, in the campus use case, this behavior was simulated by recommending that the users deactivate the system for a couple of hours by chance.

Table 3 presents an overview of contact characterization across various metrics, including distance, duration, and the number of contacts per user. Notably, the average distance between individuals during these interactions was less than 200, typically considered risky proximity. However, the high standard deviation underscores significant variations in distance values around the mean. The presence of zero as a minimum distance indicates instances where users were so close that the distance became practically negligible.

Moving on to contact duration, the average duration of these contacts stood at approximately five minutes. Nonetheless, a gap existed between the minimum and maximum contact duration values. Instances of zero duration imply that certain contacts merely involved the receipt of a beacon signal from the other person. Finally, on average, each individual had contact with approximately six others, taking into account all received beacons.

To analyze the distance and duration of contacts, Figure 8 displays the cumulative distributions of the average distance, contact duration, and the relation between these two metrics. Where Figure 8(a) demonstrates that around 60 of the

Table 3. Contact data characterization

Data	Avg	Sd	Min	Max
Distance ()	152,918	145,75	0,0	713,0
Duration (min)	5,085	7,933	0,0	55,203
Constant contacts per user	5,879	5,127	0	21

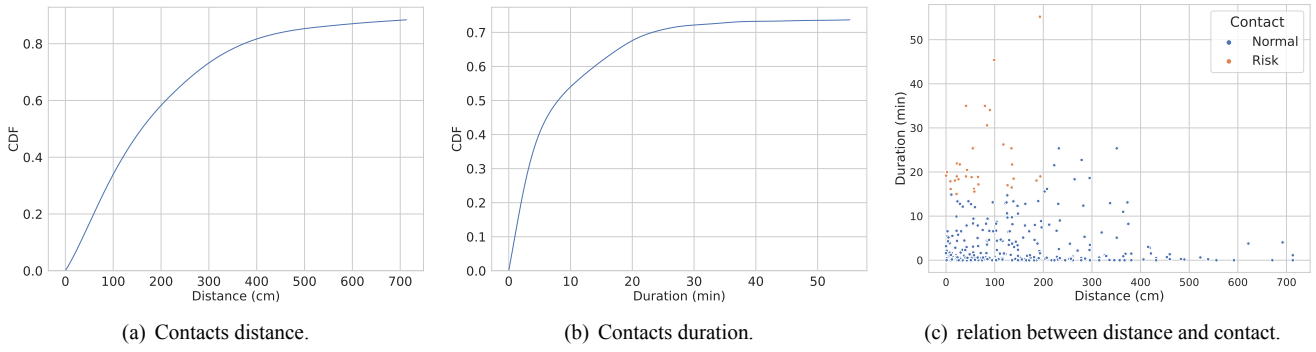


Figure 8. Quantitative analysis of the data gathered during two weeks.

recorded contacts occurred at an average distance of up to 2, while just over 30 were less than 1. Figure 8(b) shows that around 72 of contacts lasted up to 30 minutes. Additionally, a significant portion of contacts lasted up to 10 minutes, as evidenced in the distribution shown in Figure 8(b), where 54 of the contacts lasted up to 10 minutes.

Finally, Figure 8(c) presents the relation between the distance and duration of each contact. Risk Contacts, depicted as orange dots, represent the contacts with a duration greater than 15 minutes and closer than 2 meters. The risk contacts were a minority compared with the normal contacts in blue.

4.2.3 Social analysis

Figure 9(a) shows the global graph containing the volunteers’ interactions over the two weeks. Note that two individuals can have multiple contacts during a time interval, and we consider the average distance and the longest duration of contact in this graph. To better understand the contact graph, Figure 9(b) depicts the cumulative distribution function of the vertices degree, representing how many contacts each volunteer had. Therefore, we can observe that around 67 of the graph’s vertices have up to 8 edges, meaning they had contact with up to 8 different individuals.

Now, by adding the weights to the contact graphs, we can analyze different aspects as shown in Figure 10. In Figure 10(a), a dashed blue edge indicates contact with an average distance greater than 200, while black edges represent contacts with distance less than 200. Similarly, in Figure 10(b), the dashed edges represent contacts with a duration of less than 15minutes, while the black edges represent contacts with a duration of more than 15minutes. Indeed, the risk emerges when duration and distance are evaluated together. Therefore, Figure 10(c) presents the contact graph in which the black edges represent contacts where the duration was

greater than or equal to 15minutes, and the average distance was less than 2. This result allows visualizing individuals who could be notified if one contact reports a positive diagnosis.

As mentioned, two volunteers were randomly selected to report a positive diagnosis. Therefore, Figure 11 displays the infected vertices in red and the at-risk vertices that should be notified in yellow. Observe that the server only has access to the *UUIDs* of the contact nodes, and in a real-world scenario, it would not be possible to obtain the identities of the vertices that have reported positive for COVID-19.

5 Discussion and Limitations

The proposed system identified and notified users who had prolonged contact with those who reported a positive diagnosis. Furthermore, we analyzed contacts with a high potential of spreading the virus and the user’s social behavior without revealing the users’ personal information. We validate the system on a university campus, where students are usually closer to each other. However, we could identify challenges that must be addressed to improve the system’s reliability and user adoption. These challenges and possible solutions are discussed in the following sections.

5.1 User perspective

Users were instructed to use the application, preferably when other users were using it. However, it was expected that this would not always be feasible, potentially resulting in missed contacts not being collected by the system. A partial solution to this issue is to scale up the number of volunteers trying to fill these gaps statically. Additionally, volunteers may have difficulties remembering to activate tracking for certain hours of the day. In summary, recruiting individuals to participate posed one of the challenges, primarily motivating them to use the application and ensuring its reliability. An extreme solution would be the compulsory use of the apps. However, engaging users to adopt the solution while ensuring their privacy would be a better strategy to address this issue.

5.2 System perspective

The first issue was that the distance estimation based on the Bluetooth signal captured by the devices might be influenced

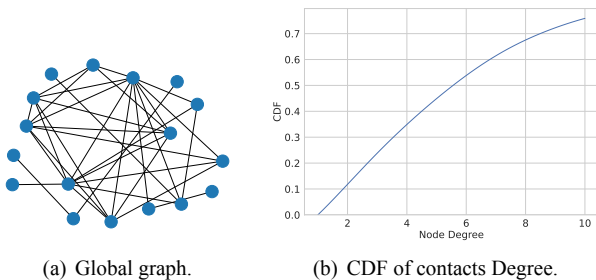
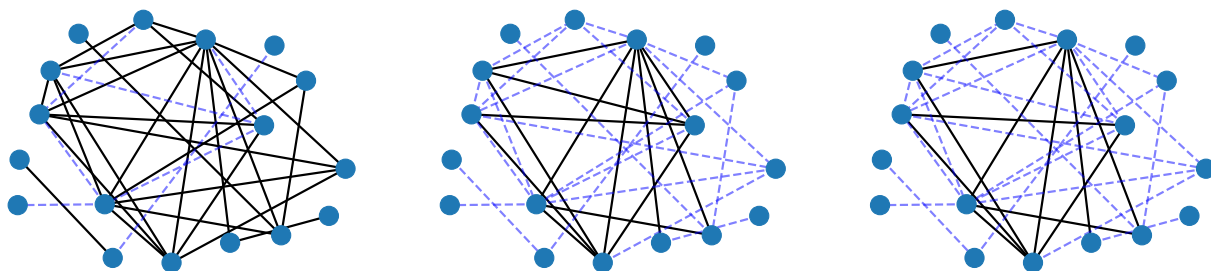


Figure 9. Contact graph and the CDF of the node degrees.



(a) Contact distance graph (Blue dashed contacts > 200 , Solid Black ≤ 200). (b) Contact duration graph (Blue dashed contacts $\leq 15\text{min}$, Solid Black $> 15\text{min}$). (c) Contact graph weighted by distance and duration simultaneously.

Figure 10. Weighted Contact graphs: (a) distance-weighted, (b) duration-weighted, and (c) distance-duration-weighted.

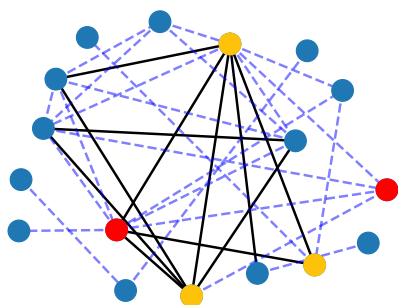


Figure 11. Infected nodes (red) and notified nodes (yellow).

by various factors beyond the distance, such as objects blocking the signal and walls dividing rooms. Additionally, other wireless signals operating on the same frequency can cause interference, further attenuating the Bluetooth signal. Furthermore, different devices transmit Bluetooth signals with varying power levels, which may require calibration.

Another aspect is the trust in the anonymity. The system relies on the user notifying a positive case, and malicious users could insert fake information about his/her diagnoses. An easy-to-fix solution to this issue would be verifying the diagnosis with a reliable database, such as a hospital or laboratory. However, personal information would be revealed and associated with the user's unique identifier in such a case, requiring additional security measures.

Moreover, malicious users can attempt to de-anonymize the users, correlating their unique ID (*UUID*) with a specific individual. Decentralized architecture systems could be de-anonymized easily, while centralized architecture systems can be more challenging, but still possible, considering the Papparazi attack and the Orwell Attack [Ahmed *et al.*, 2020; Avitabile *et al.*, 2020]. Both attacks require passive BLE devices, attempting to know every contact in the network and reconstructing the *UUID* after receiving an alert. In our proposed solution, each contact at risk receives an alert, which is scheduled and asynchronous, without any disclaimer about who is infected. The disclosure of information to a user could happen if devices have just one contact in their lifetime. This way, it would know who has a positive test after receiving an alert.

5.3 Maintenance and traffic cost

Many devices utilize either Android or iOS as their Operating System (OS), but the variety of OS versions poses a significant challenge for ensuring compatibility with mobile apps. To address this issue, we propose an open-source solution specifically tailored for Android devices. By making our solution open-source, we invite contributions from the community, which could foster sustainability in the development of other operating systems.

Regarding transmission costs. The cloud-based server application operates by receiving contact lists and identifying potentially risky contacts. Assuming that each contact tuple consumes roughly 150 bytes of data, and within a 30-minute interval, a mobile device can transmit up to 60 contacts. In the worst-case scenario, with all users transmitting full contact lists, the traffic is bounded by $(N \times 9000)$ bytes, where N represents the number of devices. This estimation assumes the transmission of raw data and could be further optimized by compacting the contact lists.

6 Conclusions

In this work, we proposed and developed a privacy-based anonymous contact tracing to preserve users' privacy so that they feel safe when using it. The proposed system detects contacts in the vicinity through beacon BLE, estimating the distance and the duration of contact between two individuals. Periodically, each device transfers the list of contacts to a cloud service. When an individual reports a positive COVID-19 case, our system promptly notifies all close contacts from the preceding 15 days who were within 2 meters for longer than 15 minutes. We implemented a mobile application and tested it on a university campus to assess its feasibility and scalability. Despite maintaining anonymity, our system logs enable detailed analysis of contact networks, providing crucial insights into disease transmission dynamics.

Anonymity is achieved through a unique identifier (UUID) generated by a curve elliptic public-private key algorithm. Devices subscribe to the MQTT topic with its UUID and exchange beacons with this UUID with other devices. In this way, the system avoids the transmission and storage of any information that could identify users or even those who reported a positive diagnosis. On the other hand, the system

allows researchers to analyze the types of contacts collected and the potential risk of contagion through tracking logs.

As future work, we envision that the proposed system could be adapted to handle other situations requiring contact tracing. For instance, in the case of a pandemic caused by different diseases, developers can adapt the risk factor and the front end of the mobile app and use it. There is also room for other solutions, such as tracking of objects. In this case, instead of notifying a positive diagnostic, a user could indicate losing an object/device.

Declarations

Acknowledgements

This work was carried out with the support of the Coordination for the Improvement of Higher Education Personnel – Brazil (CAPES) – Financing Code 001, FAPESP/MCTIC/CGI.br PORVIR-5G (#2020/05182-3 and #2023/00148-0), CNPq, and FAPES (#2022/ZQX6F, #2021/GL60J, and #2022/NGKM5).

Authors' Contributions

MFS contributed to the conception, development, and deployment of this study. BPS and PHLR contributed to the analysis of the result and writing. VFMS is the main contributor and writer of this manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

A GitHub containing the back-end and the datasets gathered during our experiments is available on <https://github.com/lprm-ufes/contact-tracing>. The mobile application source code is available on <https://github.com/lprm-ufes/contact-tracing-mobile>.

References

- Ahmed, N., Michelin, R. A., Xue, W., Ruj, S., Malaney, R., Kanhere, S. S., Seneviratne, A., Hu, W., Janicke, H., and Jha, S. K. (2020). A survey of covid-19 contact tracing apps. *IEEE access*, 8:134577–134601. DOI: 10.1109/ACCESS.2020.3010226.
- Ali, Y. and Khan, H. U. (2023). A survey on harnessing the applications of mobile computing in healthcare during the covid-19 pandemic: Challenges and solutions. *Computer Networks*, 224:109605. DOI: 10.1016/j.comnet.2023.109605.
- AltBeacon.org (2014). Altbeacon protocol specification v1.0. Available at: <https://github.com/AltBeacon/spec>.
- Apple and Google (2023). Privacy-preserving contact tracing. Available at: <https://covid19.apple.com/contacttracing>.
- Avitabile, G., Botta, V., Iovino, V., and Visconti, I. (2020). Towards defeating mass surveillance and sars-cov-2: The pronto-c2 fully decentralized automatic contact tracing system. *Cryptology ePrint Archive*. DOI: 10.1109/MIC.2022.3213870.
- Bay, J., Kek, J., Tan, A., Hau, C. S., Yongquan, L., Tan, J., and Quy, T. A. (2020). Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders. *Government Technology Agency-Singapore, Tech. Rep*, 18. Available at: https://bluetrace.io/static/bluetrace_whitepaper-938063656596c104632def383eb33b3c.pdf.
- Canetti, R., Trachtenberg, A., and Varia, M. (2020). Anonymous collocation discovery: Harnessing privacy to tame the coronavirus. *arXiv preprint*. DOI: 10.48550/arXiv.2003.13670.
- Castelluccia, C., Bielova, N., Boutet, A., Cunche, M., Lauradoux, C., Le Métayer, D., and Roca, V. (2020). ROBERT: ROBust and privacy-presERving proximity Tracing. Available at: <https://inria.hal.science/hal-02611265>.
- Cho, H., Ippolito, D., and Yu, Y. W. (2020). Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs. *arXiv preprint*. DOI: 10.48550/arXiv.2003.11511.
- Danquah, L. O., Hasham, N., MacFarlane, M., Conteh, F. E., Momoh, F., Tedesco, A. A., Jambai, A., Ross, D. A., and Weiss, H. A. (2019). Use of a mobile application for Ebola contact tracing and monitoring in northern Sierra Leone: a proof-of-concept study. *BMC infectious diseases*, 19(1):1–12. DOI: 10.1186/s12879-019-4354-z.
- Duan, S. X. and Deng, H. (2022). Exploring privacy paradox in contact tracing apps adoption. *Internet Research*, 32(5):1725–1750. DOI: 10.1108/INTR-03-2021-0160.
- Elavarasan, R. M. and Pugazhendhi, R. (2020). Restructured society and environment: A review on potential technological strategies to control the COVID-19 pandemic. *Science of the Total Environment*, 725:138858. DOI: 10.1016/j.scitotenv.2020.138858.
- Fahlani, A. A., Payer, M., and Aminifar, A. (2023). DP-ACT: Decentralized Privacy-Preserving Asymmetric Digital Contact Tracing. In *24th Privacy Enhancing Technologies Symposium, PETS 2024*. DOI: 10.56553/popets-2024-0019.
- Gov, I. (2020). Aarogya setu mobile app. Available at: <https://www.mygov.in/aarogya-setu-app/>.
- Gupta, N. K. (2016). *Inside Bluetooth low energy*. Artech House. Book.
- Gupta, R., Bedi, M., Goyal, P., Wadhwa, S., and Verma, V. (2020). Analysis of covid-19 tracking tool in india: Case study of aarogya setu mobile application. *Digital Government: Research and Practice*, 1(4):1–8. DOI: 10.1145/3416088.
- Gvili, Y. (2020). Security analysis of the COVID-19 contact tracing specifications by Apple Inc. and Google Inc. *Cryptology ePrint Archive*. Available at: <https://eprint.iacr.org/2020/428.pdf>.
- Hossmann, T., Spyropoulos, T., and Legendre, F. (2011).

- A complex network analysis of human mobility. In *2011 IEEE conference on computer communications workshops (INFOCOM WKSHPs)*, pages 876–881. IEEE. DOI: 10.1109/INFCOMW.2011.5928936.
- Jiang, T., Zhang, Y., Zhang, M., Yu, T., Chen, Y., Lu, C., Zhang, J., Li, Z., Gao, J., and Zhou, S. (2022). A survey on contact tracing: the latest advancements and challenges. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 8(2):1–35. DOI: 10.1145/3494529.
- Juneau, C.-E., Briand, A.-S., Collazzo, P., Siebert, U., and Pueyo, T. (2023). Effective contact tracing for COVID-19: A systematic review. *Global Epidemiology*, page 100103. DOI: 10.1016/j.gloepi.2023.100103.
- Lee, E., Park, K., Park, D. J., Kim, J., and Jo, C. (2021). Locally testable privacy-preserving contact tracing protocol without exposing secret seed. In *IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–5. DOI: 10.1109/ICCE50685.2021.9427587.
- Leung, K. Y., Metting, E., Ebbers, W., Veldhuijzen, I., Andeweg, S. P., Luijben, G., de Bruin, M., Wallinga, J., and Klinkenberg, D. (2024). Effectiveness of a COVID-19 contact tracing app in a simulation model with indirect and informal contact tracing. *Epidemics*, 46:100735. DOI: 10.1016/j.epidem.2023.100735.
- Li, X., Wu, W., and Chen, T. (2024). Blockchain-Driven Privacy-Preserving Contact-Tracing Framework in Pandemics. *IEEE Transactions on Computational Social Systems*. DOI: 10.48550/arXiv.2202.09407.
- Liu, M., Zhang, Z., Chai, W., and Wang, B. (2023). Privacy-preserving COVID-19 contact tracing solution based on blockchain. *Computer standards & interfaces*, 83:103643. DOI: 10.1016/j.csi.2022.103643.
- McLachlan, S., Lucas, P., Dube, K., Hitman, G. A., Osman, M., Kyrimi, E., Neil, M., and Fenton, N. E. (2020). Bluetooth Smartphone Apps: Are they the most private and effective solution for COVID-19 contact tracing? *arXiv preprint*. DOI: 10.48550/arXiv.2005.06621.
- Michael, K. and Abbas, R. (2020). Behind COVID-19 contact trace apps: The Google–Apple partnership. *IEEE Consumer electronics magazine*, 9(5):71–76. DOI: 10.1109/MCE.2020.3002492.
- Morio, K., Esiyok, I., Jackson, D., and Künnemann, R. (2023). Automated security analysis of exposure notification systems. In *USENIX Security Symposium*, pages 1–18. USENIX Association. Available at: <https://www.usenix.org/conference/usenixsecurity23/presentation/morio>.
- National Human Rights Commission of Korea (2020). Nhrck chairperson’s statement on excessive disclosure of private information of covid-19 patients. Available in <https://www.humanrights.go.kr/site/program/board/basicboard/view?boardtypeid=7003&boardid=7605315&menuid=002002001>, Last access 02/06/2022.
- Rivest, R. L., Callas, J., Canetti, R., Esvelt, K., Gillmor, D. K., Kalai, Y. T., Lysyanskaya, A., Norige, A., Raskar, R., Shamir, A., et al. (2020). The PACT protocol specification. *Private Automated Contact Tracing Team, MIT, Cambridge, MA, USA, Tech. Rep. 0.1*. Available at: <https://pact.mit.edu/wp-content/uploads/2020/04/The-PACT-protocol-specification-ver-0.1.pdf>.
- Rizi, A. K., Keating, L. A., Gleeson, J. P., O’Sullivan, D. J., and Kivelä, M. (2024). Effectiveness of contact tracing on networks with cliques. *Physical Review E*, 109(2):024303. DOI: 10.48550/arXiv.2304.10405.
- Smith, P., Sarkar, S., Patwari, N., and Kasera, S. (2024). On Passive Privacy-Preserving Exposure Notification Using Hash Collisions. *IEEE Internet of Things Journal*. DOI: 10.1109/JIOT.2024.3353255.
- Stevens, H. and Haines, M. B. (2020). Tracetogogether: pandemic response, democracy, and technology. *East Asian Science, Technology and Society: An International Journal*, 14(3):523–532. DOI: 10.1215/18752160-8698301.
- Stutzman, F. and Hartzog, W. (2012). Obscurity by design: An approach to building privacy into social media. Available at: <https://ssrn.com/abstract=2284583>.
- Trieu, N., Shehata, K., Saxena, P., Shokri, R., and Song, D. (2020). Epione: Lightweight contact tracing with strong privacy. *arXiv preprint*. DOI: 10.48550/arXiv.2004.13293.
- Troncoso, C., Payer, M., Hubaux, J.-P., Salathé, M., Larus, J., Bugnion, E., Lueks, W., Stadler, T., Pyrgelis, A., Antonioli, D., et al. (2020). Decentralized privacy-preserving proximity tracing. *arXiv preprint*. DOI: 10.48550/arXiv.2005.12273.
- Vaudenay, S. (2020). Centralized or decentralized? the contact tracing dilemma. *Cryptology ePrint Archive, Paper 2020/531*. Available at: <https://eprint.iacr.org/2020/531> Last access in 06/02/2023.
- Wahid, M. A., Bukhari, S. H. R., Daud, A., Awan, S. E., and Raja, M. A. Z. (2023). Covict: an iot based architecture for covid-19 detection and contact tracing. *Journal of Ambient Intelligence and Humanized Computing*, 14(6):7381–7398. DOI: 10.1007/s12652-022-04446-z.
- World Health Organization (2020). Contact tracing in the context of covid-19. Available at: https://apps.who.int/iris/bitstream/handle/10665/332049/WHO-2019-nCoV-Contact_Tracing-2020.1-eng.pdf Last access 27/03/2023.