A Privacy-Preserving Mobile and Fog Computing Framework to Trace and Prevent COVID-19 Community Transmission

Md Whaiduzzaman, *Member, IEEE*, Md. Razon Hossain, Ahmedur Rahman Shovon, Shanto Roy, Aron Laszka, Rajkumar Buyya, *Fellow, IEEE*, and Alistair Barros

***Abstract*—To slow down the spread of COVID-19, governments worldwide are trying to identify infected people, and contain the virus by enforcing isolation, and quarantine. However, it is difficult to trace people who came into contact with an infected person, which causes widespread community transmission, and mass infection. To address this problem, we develop an e-government Privacy-Preserving Mobile, and Fog computing framework entitled PPMF that can trace infected, and suspected cases nationwide. We use personal mobile devices with contact tracing app, and two types of stationary fog nodes, named Automatic Risk Checkers (ARC), and Suspected User Data Uploader Node (SUDUN), to trace community transmission alongside maintaining user data privacy. Each user's mobile device receives a Unique Encrypted Reference Code (UERC) when registering on the central application. The mobile device, and the central application both generate Rotational Unique Encrypted Reference Code (RUERC), which broadcasted using the Bluetooth Low Energy (BLE) technology. The ARCs are placed at the entry points of buildings, which can immediately detect if there are positive or suspected cases nearby. If any confirmed case is found, the ARCs broadcast pre-cautionary messages to nearby people without revealing the identity of the infected person. The SUDUNs are placed at the health centers that report test results to the central cloud application. The reported data is later used to map between infected, and suspected cases. Therefore, using our proposed PPMF framework, governments can let organizations continue their economic activities without complete lockdown.**

***Index Terms*—COVID-19, Community Transmission, Contact Tracing, Mobile App, Data Privacy, Fog Computing.**

Manuscript received June 10, 2020; revised August 27, 2020; accepted September 16, 2020. Date of publication September 23, 2020; date of current version December 4, 2020. This work was supported by the Australian Research Council Discovery Project: DP190100314, "Re-Engineering Enterprise Systems for Microservices in the Cloud." *(Corresponding author: Md Whaiduzzaman.)*

Md Whaiduzzaman and Alistair Barros are with the Queensland University of Technology, Queensland, Australia (e-mail: wzaman@juniv.edu; alistair.barros@qut.edu.au).

Md. Razon Hossain and Ahmedur Rahman Shovon are with the Jahangirnagar University, Dhaka, Bangladesh (e-mail: hossainmdrazon@gmail.com; shovon.sylhet@gmail.com).

Shanto Roy and Aron Laszka are with the University of Houston, TX USA (e-mail: shantoroy@ieee.org; laszka.aron@gmail.com).

Rajkumar Buyya is with the University of Melbourne, Australia (e-mail: rbuyya@unimelb.edu.au).

Digital Object Identifier 10.1109/JBHI.2020.3026060

I. Introduction

THE novel corona virus disease in 2019 (COVID-19) has spread rapidly worldwide in a short duration. It caused a significant public health crisis worldwide, and by 15 August, 2020 (i.e., within the eight months of its first infection detection), over 20.95 million persons were infected, and over 760 thousands have died [1]. Therefore, governments worldwide seek solutions to minimize the infected cases from the COVID-19 pandemic by employing mobile application based contact tracing [2], [3]. Mobile apps can help trace both infected and suspected cases in almost real-time, and governments are rushing towards developing and deploying such applications and frameworks. However, several applications raise significant privacy issues as they collect sensitive and personally-identifiable data from users, and lack user control and transparency in data processing or usage [2]–[5].

Governments are enforcing temporary lockdowns of cities to slow down the spread of COVID-19, causing tremendous economic losses. However, we can alleviate economic impact by avoiding wide-scale lockdowns and performing more targeted isolation. Therefore, introducing fog computing in economic zones (e.g., shopping malls, organization buildings) can ensure continued economic activities by alerting nearby people while mobile computing (mobile apps) can help to trace the infected and suspected cases. However, to the best of our knowledge, there is no integrated fog computing framework alongside contact tracing mobile apps that allows tracing community transmission while preserving users' data privacy. Therefore, we introduce the following research questions to find an appropriate solution to the cause.

- Q1) **Background and Issues:** What are the issues and privacy concerns in existing contact tracing apps?
- Q2) **Mobile and Fog Computing:** How to utilize mobile and fog computing to trace and prevent COVID-19 community transmission?
- Q3) **Privacy-preserving Framework:** How to develop an automated privacy-preserving e-government framework?

We answer the first question by looking into the background and issues of existing application frameworks as well as user data privacy concerns (Section II). We find that there are several mobile application frameworks developed by governments and

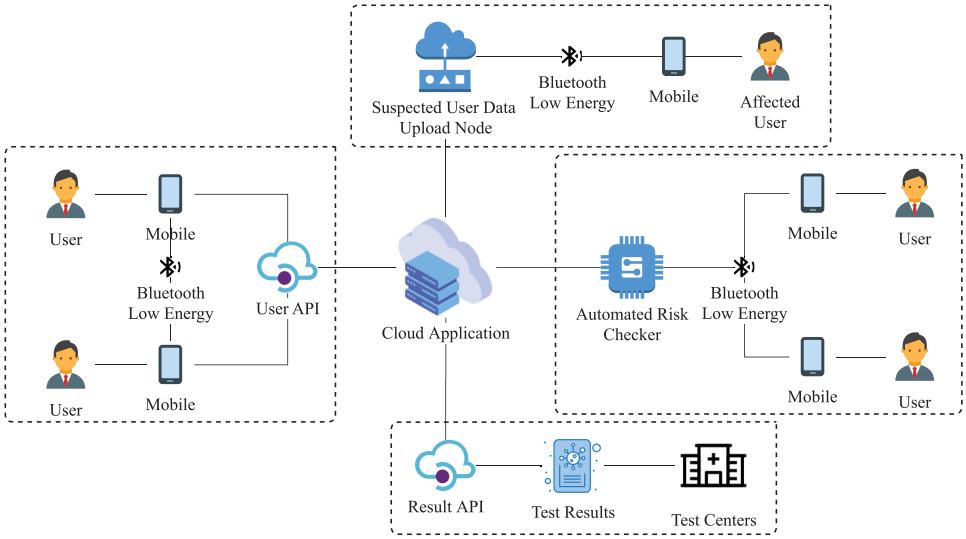


Fig. 1. System overview.

third parties to trace the COVID-19 community transmission. However, most of these applications and frameworks have failed to ensure user data privacy and suffer from other issues, such as mandatory use of apps, excessive data gathering, questionable transparency of source codes and data flow, unnecessary data usage or processing, and lack of user control in data deletion.

We answer the second question by presenting the design considerations, architecture, and workflow of our e-government application framework that utilizes mobile and fog computing (Section III). The system consists of two types of fog nodes (ARC and SUDUN), several RESTful APIs, and a central application. Users can register themselves using User API. The ARC is used to check the risk of users visiting any public places (e.g., shopping mall, organization building). Test centers send the COVID-19 test results using Result API to the central application. If the test result is positive, then the user is requested to upload locally stored contact tracing information to the cloud using SUDUN or directly from mobile app. Fig. 1 presents an overview of the proposed system.

We answer the third question by discussing the implementation overview and user data privacy solutions in our framework (Section IV). Here, we discuss the framework development based on the Amazon Web Services (AWS) solutions. Then we present the privacy preservation based on user control (voluntary, compliance, and user consent), minimal data collection (mobile number, postal code, and age group), data destruction at user's will, transparency (open source codes, clean data flow), and limited further usage of data.

Scope: We intend our framework to be deployed and controlled by the central government of a country. Governments have access to the test results and can control such an integrated mobile-fog computing framework. Moreover, relying on a private entity to manage such a framework can limit the preservation of data privacy. Additionally, in this work, we primarily consider and focus on user data privacy issues. Regardless of building a standard and secure data processing framework, we do not discuss advanced security threats related to mobile, fog, and cloud layer as there are rich literature on existing security measures [6]–[8].

Outline: The rest of the paper is organized as follows: Section II introduces the privacy and general issues of existing contact tracing frameworks. Section III discusses the design considerations, system components, and workflow of our proposed e-government framework. Section IV presents the implementation overview and privacy-oriented solutions of the framework. Finally, Section V discusses a few related privacy-preserving frameworks followed by a conclusion.

II. Background

Contact tracing and isolation of cases are required to control infectious disease outbreaks, and the consequence depends mainly on government actions and citizen responses [9]. Manual tracing is difficult and time-consuming in a pandemic situation, and governments are pushing for digital surveillance to contain the spread of the virus in the context of health informatics [10]. However, these mobile apps have raised questions over user data privacy since digital surveillance involves location tracking, limits individual freedom, and expose confidential data [5].

A. Contact Tracing

Governments can stop transmission of COVID-19 if they identify cases and their contacts quickly and get them to limit their connections with other people. Cases should isolate themselves as long as they are infectious- for at least 10 days after they became ill. Contacts must quarantine for 14 days after the last contact with an infectious patient [11]. Some cases may have close contact with many people because of where they have been or where they live, and these situations should immediately be reported.

Contact tracing might be difficult as an infected case may not remember all people who came in their contacts. Additionally, an infected person may not know or remember the phone numbers and address of their contacts. It may take longer for authorities

as well due to the required time to identify and get in touch with contacts. Therefore, mobile-based contact tracing might help track several contacts and determine who is at highest risk for infection.

Contact tracing usually requires three primary steps: contact identification, listing, and follow-up [5]. People with contact with an infected person are considered as suspected cases if they were in proximal range for more than 15 minutes, within 6 feet [12] or the distance of more than 6 feet, but stayed nearby for an hour [13].

B. Contact Tracing Frameworks and Applications

Due to the necessity of tracing infected or suspected cases, governments around the world have developed several tracing applications and frameworks [4]. The most common features of these applications are live maps and news updates of confirmed cases, location-based tracking and alerts, quarantine and isolation monitoring, direct or indirect reporting, self-assessment, and COVID-19 education [2]. Some governments involved third parties to develop such applications and encouraged citizens to use these applications.

1) Privacy Concerns: Since governments have been rushing to build tracing applications, the least they have considered about user data privacy. In most cases, users can be monitored and tracked in real-time without user's consent. If such mobile applications store the location history, user movement can be traced as well [14]. Apart from location tracking, there are several other user privacy issues, such as excessive data collection, obscure data flow, lack of user control, and data usage policies.

Abeler *et al.* suggested that we can achieve contact tracing and data protection at the same time by minimizing data processing in the existing frameworks [15]. However, many applications are not following such solutions and Howell *et al.* has suggested five primary privacy concerns for COVID-19 application frameworks [16].

- *Voluntary or Mandatory:* It should be a voluntary act whether users download and use such tracing apps. With the growing concern over data privacy, unnecessary data collection, location tracking, and other issues, users must have free wills to decide. The government or any third party cannot mandate users to use these apps in any circumstances.
- *Data Usage Limitation:* People are concerned over the collected data usage for personal safety reasons [17]. Therefore, the collected data must have usage limitations. For example, tracing data can only be used for public health and safety. Traced data cannot be used for any other purpose, e.g., law enforcement.
- *Data Destruction:* Mobile applications or a framework should automatically delete user records after a particular period [18] (e.g., usually 14-21 days and no longer than 30 days). Otherwise, users should have manual control over data deletion from the app or the central server.
- *Minimal Data Collection:* Several applications collect excessive, unnecessary data from its users, for example, an application named "`Aarogya Setu`" requires name, phone number, age, gender, profession, and details of countries visited in the last 30 days. Also, Geo-location tracing is unnecessary alongside Bluetooth or other similar wireless technologies [19].
- *Transparency:* The entire process of data collection and usage should be transparent to preserve user privacy. Application frameworks should have publicly available policies, clear and concise data flow and database, and open-source codes for transparency. Additionally, users should have full control over their data usage. Therefore, developers must follow the *compliance* and *consent* rules (GDPR, HIPAA, CCPA, etc.) strictly [18], [20].

Apart from the privacy issues mentioned above, there are certainly other things to consider, as well. For example, the mobile application generated IDs can be breached, decrypted, and resulted in exposing user information. Additionally, applications controlled by the third party may pose a severe threat as they can misuse the collected user data. Therefore, in terms of privacy, user control over the owner's data and transparency are notable factors.

2) General Concerns: Apart from several data privacy issues, different design issues are prevailing in the contact tracing apps. Many applications require a constant internet connection while it is entirely unnecessary. In some other implementations, the user can not turn off the background service of the apps, and these apps do not feature turn-off option. The apps continue to work at random times, such as while staying at home or sleeping. Therefore, it causes battery drains too fast.

Tracing correctness depends on the distance and period of contact. Several applications stores the RUERC of a nearby user device even there is a wall between two persons. Moreover, rushing to develop such applications result in *false positive* suspected victim considerations and community transmission. Communication between multiple platforms may appear troublesome and can lead to unexpected behavior. Therefore, using Google/Apple API might be helpful to introduce Bluetooth communication between two devices of different platforms (Google Android and Apple iOS). Apart from these issues, *insecure source code*, *weak data flow and process*, and different *Bluetooth-based device attacks* can cause security hazards.

3) Google/Apple Exposure Notification API: Google and Apple announced their privacy-preserving exposure notification in April 2020 and released phase one in May 2020 [21]. The API uses BLE technology and applies different hashing algorithms to generate different keys on a specific time interval [22] to prevent wireless tracking. Infected cases upload only the daily generated keys of the past 14 days, and other users download lists of these keys of infected cases of their corresponding region.

All the key-generation and risk-level checks are performed in the user's mobile device while preserving user privacy. However, this might be a concern about the resource consumption of that particular mobile device [23]. Moreover, as the user is only uploading their keys, the authority can not notify the contacts of the infected case immediately. Some other essential functionalities, such as preventing community transmission and identifying the asymptomatic spreaders, can also be troublesome.

Implementing this technology entirely depends on the corresponding public health authorities. However, the authority is required to follow the guidelines of privacy, security, and data control rules as well as the development criteria such as file and data format of storing, uploading, and downloading keys [21], [24]. This collaborative development of the Exposure Notification ensures that both of the platforms (Android and iOS) transmit and receive similar keys, and the risk level the app calculates remains similar.

C. Existing Privacy Solutions

Several works and frameworks have been proposed to address the privacy issues so far. For example, Singapore's "`Trace-together`", Reichert *et al.*'s "`MPC Solution`" [25], Altuwaiyan *et al.*'s "`Matching Solution`" [26], and Vaudenay's "`DP3T`" solution [27]. These frameworks and mobile applications introduce the concepts of voluntary use, user control and consents, cryptographic data storage, minimum user data collection, limited data usage, and overall transparency of the tracing applications and frameworks. Further details and discussion are presented in Section V where we differentiate between these works and our work as well.

Current contact tracing apps backed by the governments have numerous design and privacy issues due to the rush for community transmission tracing. Regardless of addressing existing issues, the question remains if the governments can continue the economic activities alongside. Therefore, our integrated mobile and fog computing-based framework can solve the issues by effectively tracing community transmission while organizations can run their economic activities.

III. Framework Design

Our proposed privacy-preserving e-government framework has four major components: user mobile device and two types of fog nodes (ARC and SUDUN), and a central cloud application that integrates these nodes. Mobile Unit consists of BLE, privacy dashboard, filter algorithm, file storage, and communication service. A user mobile device advertises it's own RUERC, scans for RUERCs of other nearby devices, and save the filtered RUERCs (see Algorithm 1) in the file storage.

Fog-based IoT-healthcare provides optimization of data communication, low power-consumption, and improves efficiency in terms of cost, network delay, and energy usage [28], [29]. The BLE in the ARC and SUDUN advertises a specific predefined UERC, and the mobile unit does not save these UERCs to file storage. When the user is in proximity to ARC (hospital, shopping malls, office), BLE of the ARC receives all the RUERCs around, and the fog component checks if there is any infected or suspected case. The cloud application responds with a positive or negative result without disclosing the victim's identity. If there is a positive case, the ARC transmits another predefined UERC that signifies the risk level and alerts all the mobile units around, including the infected or suspected case, hence preserving the victim's privacy. The other fog node, SUDUN, is set either in the test centers or where the authority finds it necessary. When the mobile unit receives the predefined UERC from this fog node, it checks its privacy dashboard. If the privacy dashboard allows the mobile unit to send the file to SUDUN, the mobile unit establishes a connection with SUDUN and send the file. Then, the fog component in SUDUN transfers the file to the corresponding cloud. Here, Fig. 2 presents the detailed workflow our proposed integrated mobile and fog computing framework.

A. System Features

We introduce the following features of mobile application and fog nodes in our integrated framework:

1) *Contact Tracing:* An user gets a hashed unique reference code upon registration from the system. Every two hours, another unique reference code is generated in the application, which is being shared with nearby devices upon user consent. It ensures that the broadcast data cannot be used to trace an individual. The same hashed value will be generated in the cloud application, and it can be used to check the risk level of any individual without disclosing his or her identity with others.
2) *Self-checking:* Users can check if he or she was nearby any infected victim in the last 14 days using the mobile application. The application will upload the locally stored reference codes of the device it came in contact in the last 14 days. The cloud application will check if any of the uploaded RUERC is listed as an infected victim or suspected victim. The application will notify the user accordingly.
3) *Minimum Mobile Computation:* Our proposed framework ensures minimum computation by enabling user control to turn on or off scanning, and background services. Apart from that, the mobile application features delay broadcast by avoiding unnecessary frequency and requires minimum internet connection as users only need internet while registering and uploading data for self-check.
4) *User Data Privacy:* User personal data is not stored or shared in the system. We ensure minimum data collection and avoid user location tracking or digital surveillance. Fog nodes do not identify any infected victim.
5) *Fog Node Alerts:* To minimize community transmission, we introduce automatic risk checker in public places such as shopping malls and office buildings. As there is a chance of revealing the user identity, we introduce *time delay* and *minimum entry* of people before broadcasting simple alert messages. These messages only request users to take precautions; do not reveal any RUERC or risk radius.

B. System Components

Our framework has four major components: a mobile application that broadcasts its RUERC, and stores received RUERCs from nearby devices. The ARC checks for infected cases and broadcast alerts in organization building. The SUDUN uploads data from the device of infected cases. Finally, a central cloud application integrates all these mobile and fog nodes and manages collected data.

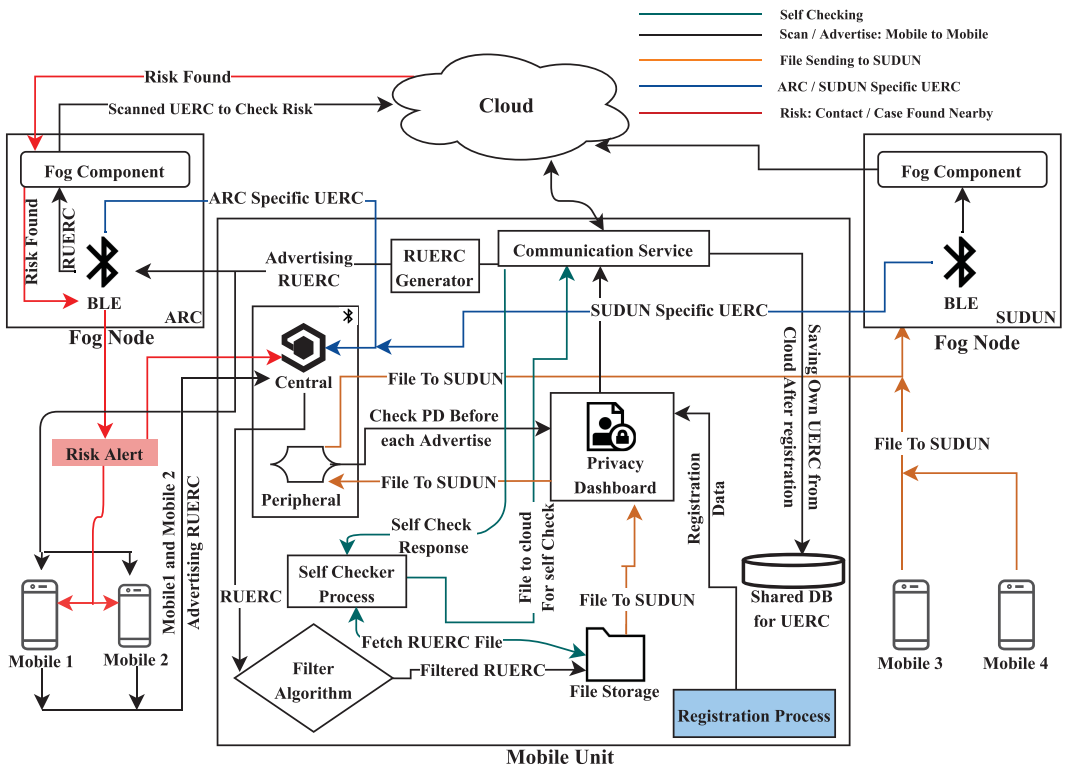


Fig. 2. Mobile and fog computing framework for tracing and preventing community transmission.

1) Mobile Application: Users download the application from the google play store or govt server and install it in the mobile device. The application broadcasts its RUERC and receives RUERC of other devices around. It also detects the special predefined UERC broadcasted by Automatic Risk Checker (ARC) and alerts the user about an infected or suspected case around.

2) Automatic Risk checker (IoT/Fog) (ARC): This fog device is set inside the hospital, shopping mall, educational institutes, and in all government-supervised organizations where there is a possibility of community transmission. The fog can receive the RUERC of the mobile devices within the Bluetooth range and interacts with cloud service to detect any infected or suspected victim. If any RUERC is found as an infected or suspected victim, it broadcasts a specific UERC throughout the place. The mobile application near the fog receives this specific UERC and alerts the user about the risk. As the fog node is registered via cloud fleet management service, the ARC node can be monitored in real-time. Thus it allows identifying any ARC node with a high rate of COVID-19 affected patient's presence. If the number of infected patients or the number of suspected patients gets higher than a predefined maximum threshold value per hour, it automatically generates emergency alert service to the organization's authority by sending emails and messages. It informs the nearby health authorities as well. The notification is broadcast only when there are at least 5 persons within the range of ARC to ensure the privacy of the infected victim. On the contrary, the notification is not sent instantly when the ARC identifies an infected or suspected victim. It notifies the users after a certain period, which ensures the identity of the infected or suspected victim is not disclosed to others.

3) Suspected User Data Uploader Node (IoT/Fog) (SUDUN): These fog nodes are similar to ARC and are set at the health centers or the COVID-19 test centers. When a test center reports a positive test result, the infected person can voluntarily allow the application to enable uploading its RUERC list from the privacy dashboard and keep the mobile phone within the range of a SUDUN. A SUDUN automatically connects with the user's device and fetches the list of stored lists of hashed RUERC from the device. Here, we enable monitoring each SUDUN using a cloud dashboard on a real-time basis. Additionally, the infected victim can also upload the RUERC list by using the mobile application and an internet connection without the help of a SUDUN.

4) Cloud/Server: The government provided secure data storage and computing server. Receiving and storing data from SUDUN, mining to predict essential patterns or super spreaders, computing to provide required information to ARC, and handling standard authentication to maintain security and privacy are its primary responsibilities.

C. System Workflow

1) User Registration: Initially, users need to register in the cloud using mobile number, age group, and postal code. The mobile number field is mandatory; however, the age group and postal code are optional. The Cloud Application generates and sends a One Time Password(OTP) to the user-submitted mobile number through SMS. Then the user enters the OTP in the mobile application, and the app sends the OTP to Cloud. The cloud application matches the user sent OTP, with the generated one.

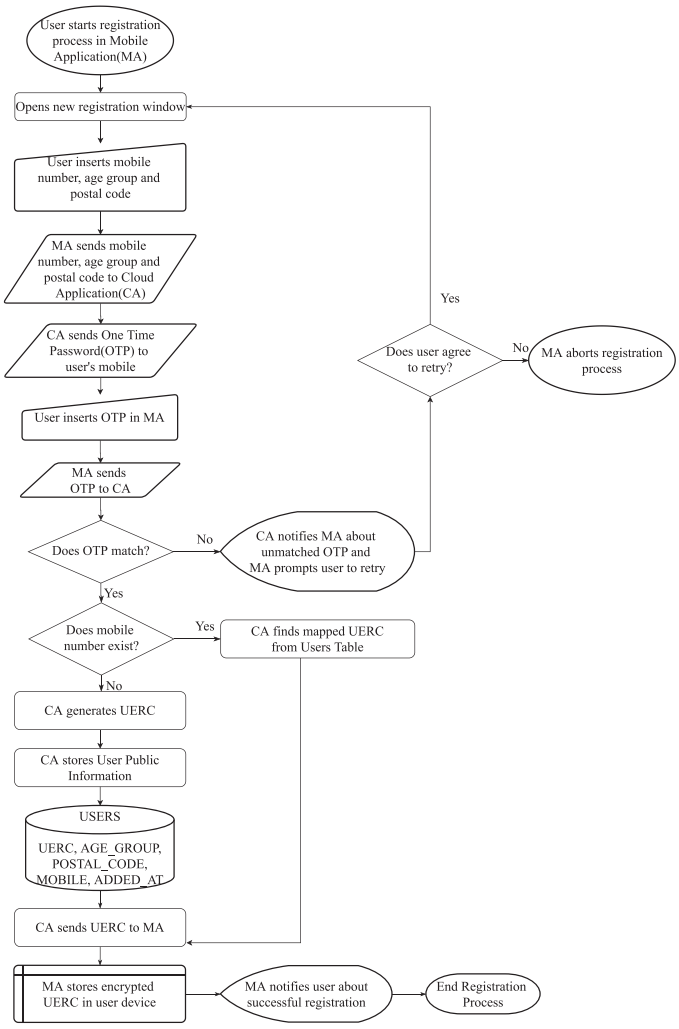


Fig. 3. User registration process.

If both matches, the cloud application generates a UERC for the user and send it back to the user installed mobile app. Finally, the mobile app stores the UERC in encrypted format in the user device and then proceeds to scan IDs broadcasted by other users. The user device itself broadcasts a rotational UERC (derived from the initial UERC) as well. Fig. 3 presents the complete registration process.

2) Key Generation: Our framework generates Rotational UERC (RUERC) with an interval of two hours. A person is considered a suspect case if he stays nearby to an infected person for an hour. As the RUERC changes every two hours, possibly the users are supposed to receive more than one RUERC if they stay nearby longer than an hour (e.g., in case they are neighbors). Our mobile application ensures that even if a person comes in close at any minute, it will store individual key and compare against the timestamp. It increases the accuracy of finding contact without a rigorous need for computation. Moreover, the RUERC enhances the privacy of the user because always broadcasting the same UERC may result in wireless tracking. While registering, the mobile unit receives a UERC from the cloud, and from this UERC, it generates RUERCs using the AES algorithm on each

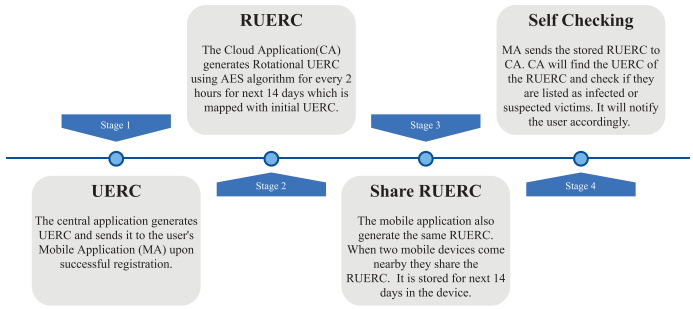


Fig. 4. Key generation steps.

two-hour interval for the next 14 days. When two devices come close, they transmit and save these RUERCs. Fig. 4 presents the steps of RUERC generation.

3) RUERC Scan Service: Scan service allows the application to run in the background. It creates a new thread so that the user does not feel any obstacle while using other mobile apps.

The user has the privilege to stop and then restart running the service at any time. Scanning and advertising RUERC, creating files, and saving data are its primary responsibilities.

Scan service uses Bluetooth Low Energy (BLE) technology to share RUERC between two mobile devices. The BLE consumes significantly less power to communicate, control, and monitor IoT devices. It uses "`central`" and "`peripherals`", which define a network called piconet [30]. Central scans for the advertisement and peripheral makes the advertisement. In our framework, the advertisement consists of the defined RUERC. The Generic Attribute Profile (GATT) is designed to send and receive short pieces of data known as "attributes," and it is built on Attribute Protocol (ATT), which uses 128-bit unique ID [30]. In our framework, to maintain the minimal data collection policy and preserve device power, we are omitting the use of GATT and ATT. We are using emphperipheral to transmit the RUERC and the *central* to scan these RUERCs. While scanning, we filter these RUERCs with our *suspect filtering algorithm* and save accordingly.

4) RSSI: The framework uses a Received Signal Strength Indicator (RSSI) to identify the distance between devices. RSSI shows the strength of the received signal. It is calculated in dB and depends on the power and chipset of the broadcasting device. It also depends on the transmitting medium. If there is any obstacle between the receiver and the sender, the signal strength will decrease, so the RSSI. Therefore, for a different manufacturer, the value of RSSI in a specific medium shows different values. However, for a particular manufacturer, RSSI shows the intensity of the signal strength. RSSI does not provide the accurate distance but using path loss model [31], an approximate distance can be found,

$$RSSI = -10 \cdot n \cdot \log_{10}(d) + C$$

Here, n is the path loss exponent that depends on the transmitting medium, d is the distance, and C is a constant.

5) RUERC Storage in Mobile Device: To maintain user privacy, the mobile application creates two files in the internal

storage of the mobile device. Even the user does not have permission to access these files. When the peripheral receives any signal, one file temporarily stores this signal information, and other stores the information of the filtered signal. When the device comes within the range of SUDUN, the signal information of the filtered file is sent to SUDUN or directly to the cloud to perform self-check.

6) Suspect Filtering: According to CDC, a distance of six feet between the case and contact is safe to maintain [12]. A person is considered a contact if he/she is within six feet for at least fifteen minutes or is within the proximity for an hour [13]. The scan service scans all the RUERCs within the range of Bluetooth. However, considering the rules mentioned above, continuous scanning is unnecessary as it causes battery and memory consumption. Therefore, the scanning service scans for 700 ms with three minutes interval. We develop our suspect filtering algorithm considering these factors so that the power and memory consumption is minimal, and identifying suspects is more accurate.

There are two types of files in the mobile device. One contains all the signal information (RUERC, distance, and timestamp) it receives from nearby devices on each interval. If any received signal passes the filtering conditions, associate information (RUERC, distance, timestamp, and duration) is stored in another file. This file is called the *final file* and sent to the cloud once the user is found COVID-19 positive. A specific RUERC is stored in the final file only once a day. Here, the suspect filtering algorithm is shown in Algorithm 1.

Algorithm 1: Filtering Suspected RUERC in Mobile Application.

```
/* T_F = Temporary File, F_F = Final File,                    */
/* T_c = Current Time, T_r = Registerd Time,                  */
/* T_15 = 15 minutes, T_60 = 60 Minutes,                      */
/* D_c = Current Distance, D_r = Registered Distance, D = Standard Distance (6 Feet), C_t = Contact, C_s = Case, S_cS_n = Scanned Signal, R_cS_n = Recorded Signal in T_F,   */
/* ASU = ARC specific UERC, RSU = Risk specific UERC,         */
/* SSU = SUDUN specific UERC                                  */
1  foreach Signal in R_cS_n do
2      if S_cS_n.contains(Signal) == false then
3          R_cS_n.remove(Signal)
4  foreach Signal in S_cS_n do
5      if Signal.RUERC == self.RUERC OR Signal.RUERC == ASU then
6          return
7      if Signal.RUERC == RSU then
8          showRiskAlert() return
9      if Signal.RUERC == SSU then
10         sendFileToSUDUN() return
11     if R_cS_n.contains(Signal) == false then
12         R_cS_n.add(Signal)
13     else
14         Set T_c = Signal.time, T_r = R_cS_n.get(Signal.RUERC).time, D_c = Signal.distance, D_r = R_cS_n.get(Signal.RUERC).distance
15         if D_c <= D AND D_r <= D then
16             if T_c − T_r >= T_15 then
17                 C_t = Signal
18                 if (F_F.contains(C_t) AND F_F.get(C_t).time.date == C_t.time.date) OR isUserConsentToStoreData == false then
19                     return
20                 F_F.add(C_t)
21             else
22                 D_r = D_c
23         else if T_c − T_r >= T_60 then
24             C_t = Signal
25             if (F_F.contains(C_t) AND F_F.get(C_t).time.date == C_t.time.date) OR isUserConsentToStoreData == false then
26                 return
27             F_F.add(C_t)
28         else
29             D_r = D_c
```

D. Data Processing

The framework is divided into three data processing layers to preserve user privacy and restrict access to data in various processes. Users can register to the system using a public communication process via mobile application. In this same access layer, the test centers can send the test results using RESTful API services through HTTPs protocol. The users can connect ARC and SUDUN fog node through the BLE protocol and send data to fog nodes. The computation and initial screening of user-uploaded data in fog nodes is done in a protected network layer. The APIs which are consumed by user mobile applications, test centers, and fog nodes are also in this layer. The main data processor consisting of a set of application nodes resides in a protected network, and users cannot access the data processor directly. The processed data is stored in data storage, which is configured in a restricted private subnet. This subnet can only be accessed from the ancestor private subnet, not from any public subnet. The data layer is protected from SQL injection as it is not directly connected through any user entries. Fig. 5 displays the process flow diagram.

IV. Framework Development

In this section, we discuss the implementation overview of our framework with regards to *Amazon Web Service*. Additionally, we discuss the database design, analysis of our contact tracing graph, and our privacy-preserving solutions.

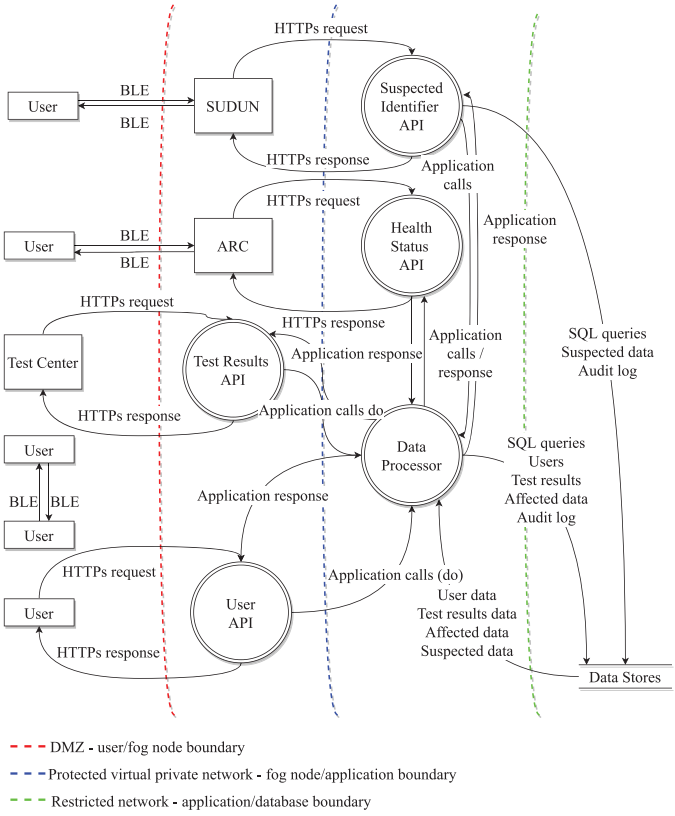


Fig. 5. Process flow diagram with privacy and security boundaries.

A. Implementation Overview

We have implemented our framework using Amazon Web Services (AWS) [32], [33]. As it is a generic framework, it can be applied in other IoT/Cloud platforms too. The fog nodes, ARC and SUDUN, consist of AWS Greengrass components and Lambda functions. They are connected to the AWS IoT Core service using the MQ Telemetry Transport (MQTT) protocol. The authenticity and security of the fog nodes are ensured by using IoT Device Defender. IoT Device Management service is used to monitor and audit the fog nodes. Incoming data from the fog nodes are passed to Simple Queue Service(SQS). The queue data is processed via Lambda functions and moved into the application containers, which is managed via the Kubernetes cluster within an auto-scaling group for dynamic scaling of the master nodes and worker nodes.

After the data is processed in the application nodes, it is sent to the Redis Cache tier and eventually to the Amazon Relational Database Service (RDS). For the administration of the cloud application, an SSH connection is provided via Elastic Load Balancing to a Bastion server. The users need to be connected to the cloud application only on the registration process via HTTPs connection to a load balancer. The test centers send the test results using RESTful API over HTTPs. The scheduled tasks, for example, generating the user's rotational UERC, is done in an Elastic Container Service, which uses the CloudWatch event rule at a specific time of the day. The application generates alarms for any irregular activities like device connection error or access failure via CloudWatch alarms to the proper authority. Users are notified from ARC, SUDUN, and cloud application via Simple Notification Service. For further communication processes to authorized organizations and administrators, the Simple Email Service is used. For restricting unauthorized access to the applications and data layers, private subnets are used. To ensure efficient data availability in the cloud, a database instance is stored in a separate availability zone. Here, Fig. 6 illustrates the implementation overview.

B. Database

The cloud database includes several tables to trace infected and suspected cases, to alert users, to prevent community transmission in organizations through ARC, and collect test results from SUDUN. The user table stores user registration information (mobile number, age group, postcode, and timestamp) and the initial UERC. As we are generating new UERCs associated with the primary user UERC, we need mapping in between those unique reference codes. We store test results (result ID, test organization ID, timestamp, test result, and user mobile number) in a separate table and extract the infected users (affected UERC, test result ID, and affected ID). Now, we map between the registered user table and affected user table to find out the suspected users and store the suspected UERC, duration, timestamp, and distance radius alongside generating a new suspect ID. We also need an organization database where we store organization information (name, email, geo-location, address.) and associated permissions for ARC and SUDUN. Fig. 7 presents the relations between all these tables.

C. Contact Tracing Graph

To identify and trace COVID-19 community transmission, we introduce and utilize a graph database called "`Neo4j`" [34] to determine the contact graph. Fig. 8 presents a portion of the contact tracing graph with synthetic data of contact tracing that indicates the implemented framework is able to trace COVID-19 community transmission. The red, blue, and green circles indicate the test centers, users, and test results. The lines present directed edges of the graph between infected and suspected victims. The tracing graph can be used to identify individuals who had come in close contact with infected victims almost in real-time. We can determine a potential super spreader as well, using the contact tracing graph. Additionally, suspected victims identified by the graph can be informed to take cautionary actions to prevent community transmission.

D. Privacy Preservation

As we discussed five major data privacy issues earlier in Section II, here, we discuss solutions to these issues in terms of implementing our framework.

1) Voluntary: Users have full control over the usage of this application and the stored data in the application storage. Even it is an e-government framework, no one can force users to use the mobile app, and the participation is voluntary. While using the mobile application, users have a privacy dashboard (Fig. 9a)

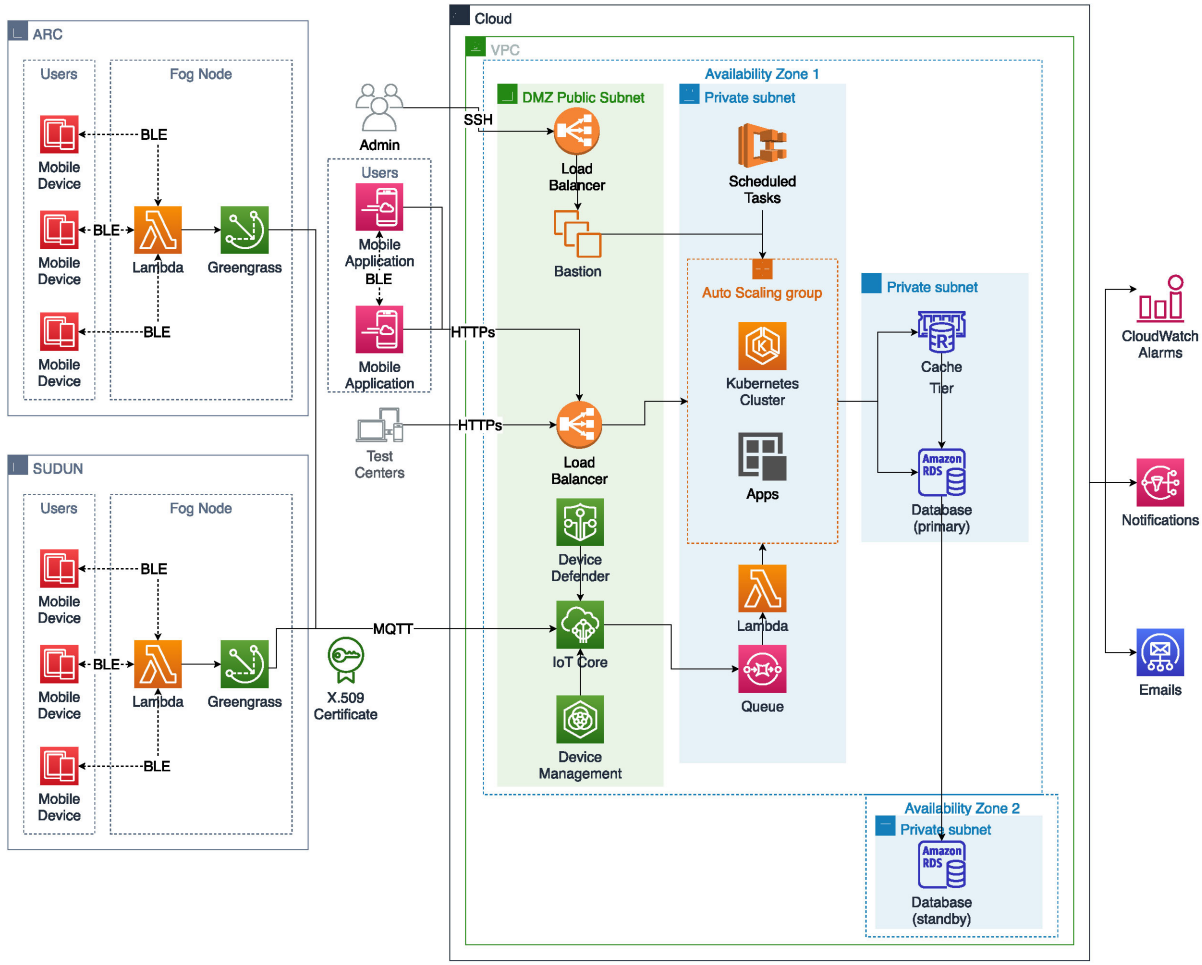


Fig. 6. Implementation overview with regards to AWS.

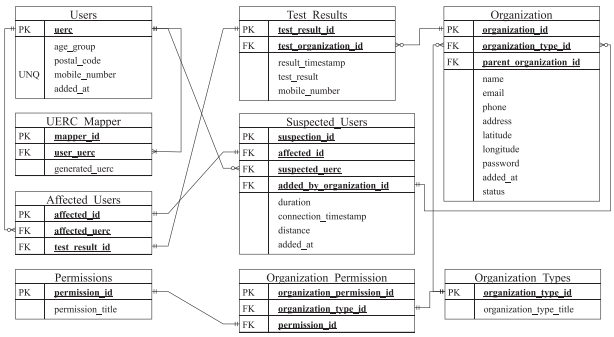


Fig. 7. The cloud database consisting of mandatory tables.

where he can allow if the app wants to store collected RUERCs in the local storage, use device Bluetooth to share user's RUERC, and if the user wants to share the collected RUERCs with the government.

2) Minimal Data Collection: Initially, the system collects the user's *age group*, *postal code*, and *mobile number* for registration and, in return, provides a unique ID (UERC) to be stored in a user's mobile device. Then the mobile application collects *RUERC*, *timestamp*, *duration*, and *distance* measure (based on RSSI) from other mobile devices that came in contact with the user's device. Age group and postal code are optional fields. These fields are used to define clusters and identifying super spreaders. The phone number is used to alert contact to take precautions. No personal information of the user is collected during or after the registration procedure. Fig. 9 presents the mobile application user interface with mandatory and non-mandatory fields.

3) Data Destruction: The mobile application deletes collected RUERCs no later than 21 days calculated from the latest timestamp. In the cloud application, UERC and associated data from the suspected table is deleted in 30 days of entry timestamp. Additionally, a user has full control over his data and can delete it manually from the application storage instantly. A user can also request for account deactivation and deletion of data from the cloud database. However, it may take up to two weeks to synchronize data in all replica databases.

4) Transparency: Transparency is vital in data privacy, and governments should make the source codes open and publish the user data flow in the whole framework. Transparency motivates users to use an application and results in better management and increased public engagement. Consent and compliance can

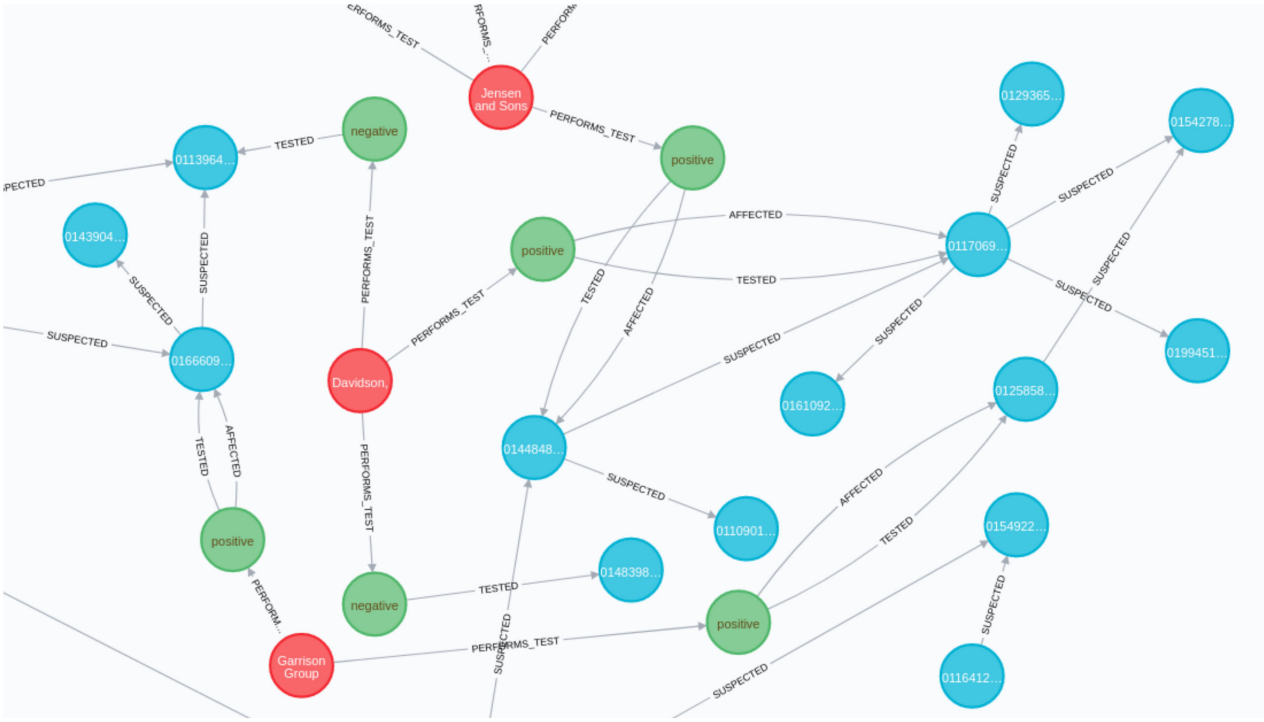
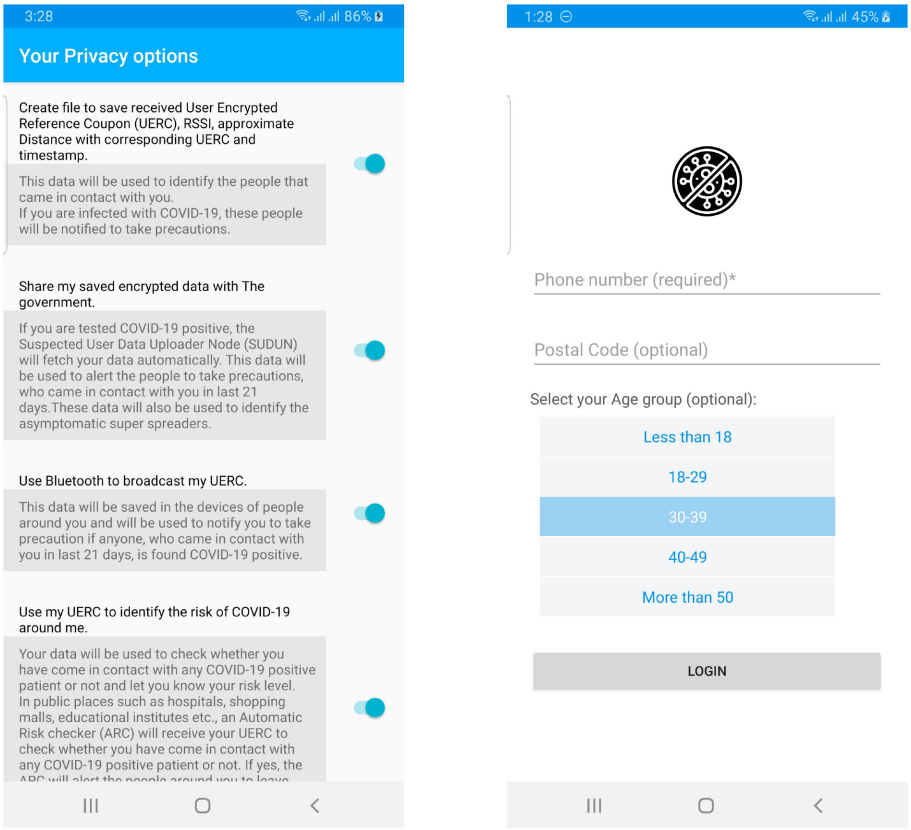


Fig. 8. Contact tracing graph.



(a) User Consent Privacy Dashboard (b) Minimal Data Collection during Registration

Fig. 9. Privacy concerned mobile application user interfaces.

TABLE I
COMPARISON BETWEEN EXISTING SOLUTIONS AND OUR SOLUTION

Features / Frameworks	Privacy-preserving Approaches					Fog Computing		Design Approaches		
	Volunatary	Data Usage Limitation	Data Destruction	Minimal Data Collection	Transparency	Risk Check	Infected/Suspected Data Upload	Temporary BLE ID	No Location Tracking	Minimal Internet Use
Tracetogether	✓	✓	✓	✓	✓	x	x	✓	✓	✓
MPC Solution	✓	✓	x	✓	x	x	x	✓	✓	x
Matching Solution	x	✓	x	x	x	x	x	✓	✓	✓
DP-3T Solution	✓	✓	✓	✓	✓	x	x	✓	✓	N/A
COVIDSafe	✓	✓	✓	✓	x	x	x	x	✓	✓
PPMF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

play a crucial role in presenting transparent data processes and improving decision making.

5) Limited Data Usage: The collected data should have limited use cases, and our framework opens the opportunity to determine a *super spreader*, and *clustered view* of positive cases. Additionally, the infected and suspected lists can help calculate the risk factors in terms of community transmission. However, while employing, further data usage should be minimal, and users need to have a clear idea of why and how data is being processed.

V. RELATED WORK

Qiang Tang discussed a few existing privacy-preserving framework solutions that attempted to solve several user data privacy issues [35]. Author provided some observations and privacy solutions on Singapore's "`Tracetogether`", Reichert *et al.*'s "`MPC Solution`" [25], Altuwaiyan *et al.*'s "`Matching Solution`" [26], and Vaudenay's "`DP3T`" solution [27].

Singapore's Tracetogether: Affected users are compelled to share with the Ministry of Health the locally saved information. In our framework, this is voluntary to share any information with the system. Additionally, there is a risk of decomposition of the app and collecting the geo-location data from the app. As we are not storing any geo-location data anywhere, this does not possess any risk of transpassing.

Reichert et al.'s MPC Solution: This solution requires the user's smart device to be used to acquire and save geo-location data, and this location trace is shared with the Health Authority (HA) [25]. This may raise a privacy issue of the users as it does not mention if they can preserve their privacy by controlling what information they want to share with the authority and what information they do not want to share. We provide a clear and concise privacy dashboard to the users so they can choose the information they want to share and alter their choices. The solution does not consider the scalability of computation of extensive data set collected from the devices, and HA requires to arrange garbled circuits (a cryptographic protocol that encrypts computation) for all infected users. We take precautionary steps in fog nodes to filter the collected data. Additionally, the central application uses an auto-scaling technique and a queue service to facilitate the upcoming data.

Altuwaiyan et al.'s Matching Solution: This solution utilizes a privacy-preserving matching protocol between users along with proper distance measuring procedure [26]. We are using similar metrics to calculate the distance between mobile devices. On the other hand, infected users' exact location is shared with the central server, which indicates severe privacy violation. We do not use individual users' locations and thus preserve users' location privacy.

The DP-3 T Solution: The DP-3 T solution provides a privacy-aware framework while considering three design considerations: low-cost design, unlinkable design, and hybrid design. The solution takes advantage of the content delivery network and provides a cost estimation per patient. Authors discussed privacy concerns such as the social graph (social relationships between users), interaction graph (physical interaction nearby), location traceability (tracing individuals), at-risk individuals (suspected cases), positive status (infected cases), and exposed location (partial identification of places a positive case visited). They also discussed the addressable privacy concerns in terms of the three design considerations.

COVIDSafe: The app collects user name, phone number, age-range, and postcode followed by the consent of the user and sends to the server. The app stores encrypted user ID, time of contact, and Bluetooth signal strength and keeps this data for 21 days. The case can upload these data voluntarily [36]. The government has amended the Privacy Act 1988 to prevent the misuse of these data [37]. The user can uninstall the app to delete these data and to delete all data from the server, the user needs to wait until the pandemic is over whereas, our framework allows users to delete all data within 14 days.

Table I presents the difference between existing mobile application frameworks and our integrated mobile-fog computing framework (PPMF). Here, we categorize different features in terms of privacy-preserving approaches (voluntary, data usage limitation, data destruction, minimal data collection, and transparency), fog-based integrated solutions (risk check, infected/suspected data upload), and general design approaches (temporary BLE IDs, no geo-location trace, and minimal internet requirements). We put our framework (PPMF) at the end of the list and find it ticks all these features.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we have presented a mobile and fog computing-based integrated framework that can trace and prevent community transmissions alongside maintaining user data privacy. In *PPMF*, we consider minimal data collection and provide a temporary RUERC in encrypted form for storing in user devices.

The *ARC* is responsible for tracing positive cases in public places and sending alerts to nearby people without revealing the user's identity. The *SUDUN* uploads data to the central database about infected and suspected cases that can be processed to identify a super spreader and visualize the clusters of cases based on postal codes and age groups.

Minimal and undetectable data collection, user control, and system transparency are essential factors to ensure user data privacy. Privacy dashboard-based on compliance and user consent makes it convenient and encourages citizens to use such a user-friendly e-government framework. Therefore, using the structure of our proposed *PPMF* framework, governments can continue their economic activities while tracing and minimizing the mass-level community transmission. In the future, we plan to develop a super spreader detection model and clustering methodology of infected cases, based on our framework.

REFERENCES

[1] "Home - Johns Hopkins coronavirus resource center," 2020. Accessed May 31, 2020. [Online]. Available: https://coronavirus.jhu.edu/

[2] T. Sharma and M. Bashir, "Use of apps in the Covid-19 response and the loss of privacy protection," *Nat. Med.*, vol. 26, no. 8, pp. 1–2, 2020.

[3] J. Chan *et al.*, "Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing," 2020, *arXiv:2004.03544*.

[4] H. Cho, D. Ippolito, and Y. W. Yu, "Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs," 2020, *arXiv:2003.11511*.

[5] R. Raskar *et al.*, "Apps gone rogue: Maintaining personal privacy in an epidemic," 2020, *arXiv:2003.08567*.

[6] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog *et al.*: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, 2018.

[7] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl*. Springer, 2015, pp. 685–695.

[8] S. Singh, Y.-S. Jeong, and J. H. Park, "A survey on cloud computing security: Issues, threats, and solutions," *J. Netw. Comput. Appl.*, vol. 75, pp. 200–222, 2016.

[9] J. Hellewell *et al.*, "Feasibility of controlling Covid-19 outbreaks by isolation of cases and contacts," *Lancet Global Health*, vol. 8. no. 4, 2020.

[10] J. Andreu-Perez, C. C. Y. Poon, R. D. Merrifield, S. T. C. Wong, and G. Yang, "Big data for health," *IEEE J. Biomed. Health Inform.*, vol. 19, no. 4, pp. 1193–1208, Jul. 2015.

[11] R. Pung *et al.*, "Investigation of three clusters of Covid-19 in Singapore: Implications for surveillance and response measures," *Lancet*, vol. 395, no. 10229, pp. 1039–1046, 2020.

[12] T. C. for Disease Control and Prevention, "Social distancing, quarantine, and isolation," May 2020. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/social-distancing.html

[13] E. Gurley, "COVID-19 Contact Tracing, Coursera, Johns Hopkins University, USA," May 2020. [Online]. Available: https://www.coursera.org/learn/covid-19-contact-tracing?edocomorp=covid-19-contact-tracing

[14] A. Berke, M. Bakker, P. Vepakomma, R. Raskar, K. Larson, and A. Pentland, "Assessing disease exposure risk with location data; a proposal for cryptographic preservation of privacy," 2020, *arXiv:2003.14412*.

[15] J. Abeler, M. Bäcker, U. Buermeyer, and H. Zillessen, "Covid-19 contact tracing and data protection can go together," *JMIR mHealth uHealth*, vol. 8, no. 4, 2020, Art. no. e19359.

[16] P. H. O'Neill, T. Ryan-Mosley, and B. Johnson, "A flood of coronavirus apps are tracking us. Now it's time to keep track of them," MIT Technology Review, May 2020. [Online]. Available: https://www.technologyreview.com/2020/05/07/1000961/launching-mittr-covid-tracing-tracker/

[17] L. Simko, R. Calo, F. Roesner, and T. Kohno, "Covid-19 contact tracing and privacy: Studying opinion and preferences," 2020, *arXiv:2005.06056*.

[18] A. Dubov and S. Shoptaw, "The value and ethics of using technology to contain the Covid-19 epidemic," *Amer. J. Bioethics*, vol. 20, no. 9, pp. 1–5, 2020.

[19] "Aarogya setu faqs," May 2020. [Online]. Available: https://static.mygov.in/rest/s3fs-public/mygov_159056978751307401.pdf

[20] A. R. Shovon, S. Roy, A. K. Shil, and T. Atik, "GSPE compliance: Implementation use cases for user data privacy in news media industry," in *Proc. 1st Int. Conf. Adv. Sci., Eng. Robot. Technol.*, 2019, pp. 1–6.

[21] Google, "Exposure notification - FAQ v1.1," May 2020. [Online]. Available: https://blog.google/documents/73/Exposure_Notification_-_FAQ_v1.1.pdf

[22] Google, "Exposure notification - Bluetooth Specification pages," May 2020. [Online]. Available: https://blog.google/documents/70/Exposure_Notification_-_Bluetooth_Specification_v1.2.2.pdf

[23] R. Damasevicius, G. Ziberkas, V. Stuikys, and J. Toldinas, "Energy consumption of hash functions," *Elektronika ir Elektrotechnika*, vol. 18, pp. 81–84, Dec. 2012.

[24] Google, "Exposure notifications service additional terms," Jun. 2020. [Online]. Available: https://blog.google/documents/72/Exposure_Notifications_Service_Additio nal_Terms.pdf

[25] L. Reichert, S. Brack, and B. Scheuermann, "Privacy-preserving contact tracing of Covid-19 patients," Cryptology ePrint Archive, Report 2020/375, 2020. [Online]. Available: https://eprint.iacr.org/2020/375

[26] T. Altuwaiyan, M. Hadian, and X. Liang, "Epic: Efficient privacy-preserving contact tracing for infection detection," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.

[27] S. Vaudenay, "Analysis of dp3t," Cryptology ePrint Archive, Report 2020/399, 2020. [Online]. Available: https://eprint.iacr.org/2020/399

[28] R. Mahmud, F. L. Koch, and R. Buyya, "Cloud-fog interoperability in iot-enabled healthcare solutions," in *Proc. 19th Int. Conf. Distrib. Comput. Netw.*, 2018, pp. 1–10.

[29] G. Yang *et al.* "Iot-based remote pain monitoring system: From device to cloud platform," *IEEE J. Biomed. Health Inform.*, vol. 22, no. 6, pp. 1711–1719, Nov. 2018.

[30] C. Gomez, J. Oller Bosch, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors (Basel, Switzerland)*, vol. 12, pp. 11734–11753, Dec. 2012.

[31] P. Kumar, L. Reddy, and S. Varma, "Distance measurement and error estimation scheme for RSSI based localization in wireless sensor networks," in *Proc. 50th Int. Conf. Wireless Commun. Sensor Networks*, 2009, pp. 1–4.

[32] D. Robinson, *Amazon Web Services Made Simple: Learn How Amazon EC2, S3, SimpleDB and SQS Web Services Enables you to Reach Business Goals Faster*. Emereo Pty Ltd, 2008.

[33] "Amazon web services (AWS) - cloud computing services," Accessed: Jun. 7, 2020. [Online]. Available: https://aws.amazon.com/

[34] "Neo4j graph platform – The leader in graph databases," Accessed: Jun. 7, 2020. [Online]. Available: https://neo4j.com/

[35] Q. Tang, "Privacy-preserving contact tracing: Current solutions and open questions," 2020, *arXiv:2004.06818*.

[36] A. Government, "Privacy policy for COVIDSafe app," Jun. 2020. [Online]. Available: https://www.health.gov.au/using-our-websites/privacy/privacy-policy-for-covidsafe-app

[37] A. Government, "Privacy amendment (Public Health Contact Information) act 2020," Jun. 2020. [Online]. Available: https://www.legislation.gov.au/Details/C2020A00044