CovidGuard - Exposure Notification Platform for Contact Tracing

Sree Ram Boyapati - a1775690
sreeram.boyapati@student.adelaide.edu.au
University of Adelaide
Adelaide, South Australia

Zach Wang
zach.wang@adelaide.edu.au
University of Adelaide
Adelaide, South Australia

Damith Ranasinghe
damith.ranasinghe@adelaide.edu.au
University of Adelaide
Adelaide, South Australia

ABSTRACT

Coronavirus Disease of 2019 (COVID-19) was classified as a pandemic in March 2020. It spreads exponentially through contact with respiratory droplets of infected patients when they cough or sneeze. It has a high incubation period before patients show any symptoms and get diagnosed. To contain the COVID-19 clusters, Contact tracing solutions have been developed to trace back the contacts of an infected user and notify potential contacts that they may have to get themselves tested. Contact tracing solutions can also be used as surveillance mechanisms; hence, a good contact tracing solution should ask for minimal personal information and ensure that tracing between the users is secure from government, hackers and telecom networks. We would like to describe the work we have done in making a custom exposure notification platform based on the invite-only Google Apple Exposure Notification (GAEN) platform.

CCS CONCEPTS

• **Social and professional topics** → *Patient privacy*; • **Networks** → **Mobile and wireless security**; *Short-range networks*; • **Hardware** → *Wireless devices*.

KEYWORDS

Google-Apple Exposure Notifications, Android Application Platform, Contact Tracing, COVID-19

1 INTRODUCTION

COVID-19 cluster network follows a power-law distribution[21] growing exponentially, and it is highly resilient to eradicate from the population. It has a high fatality rate among older patients who may be suffering from other ailments. To reduce the maximum hospitalisation rate and flatten-the-curve[29] of new infections, Social distancing measures were introduced. Contact Tracing is an efficient tool to trace epidemic clusters and contain them. It has found success in containing the EBOLA transmission in Africa[23]. However, it was paper-based, and the tracing staff had to visit the patients daily, and the entire program had around a few thousand citizens. Authorities were able to provide better care to probable users and understand the disease well by monitoring symptoms for potential infections. A case study was conducted to evaluate a mobile phone application based contact tracing[10] in Sierra Leonne. However, It was more of a CRM application to manage contact tracers rather than automated contact tracing.

COVID-19 is a global pandemic with millions of infected users with an incubation period of 5-6 days[27] during which infected users can have several contacts across the globe. The symptoms can be confused with mild or moderate pneumonia during the onset. Governments can also use the pandemic to increase the surveillance on their population and monitor journalists and activists on whom they were meeting or where they were at a particular time. The contact tracing should also support the various stages of infection based on Susceptible, Exposed, Infectious, Recovered (SEIR) epidemic model. An ideal contact tracing protocol should adhere to Health Insurance Portability and Accountability Act (HIPAA)[9] and General Data Protection Regulation (GDPR)[2] or any equivalent regulations.

To boost the adoption of the contact tracing protocol, Google and Apple have launched contact tracing capabilities[13] into their respective operating systems for use by sovereign health authorities. They have complete control over the implementation, and the latest version that is working in everybody's phone is not audited. Vaudenay and Vuagnoux [25] raised concerns on vendor lock-in and opaque server infrastructure in the analysis of Switzerland's contact tracing application, SwissCovid. I tried to address some of these in our complete end-to-end implementation.

2 RELATED WORK

In Ahmed et al. [4], Contact tracing apps are classified into three varieties -

- `Centralised Architectures` The contacts of the infected user are sent to the centralised server, and centralised server may or may not contain the personal information of all users. The contact information may be periodically uploaded from the device or in real-time based on the design of the system. Such a setup is highly efficient in the sense that health authorities can immediately notify potential contacts without waiting for consent from the infected user. Security of the platform can be scoped to only the server treating the devices as transmitters. However, there is a huge loss in privacy, and if the server is not honest-but-curious, It can be misused by authoritarian states to build a social-graph.
- `Decentralised Architectures` To preserve privacy as the top-most priority, Decentralised architectures were formulated with GAEN being the most popular platform. The server may or may not collect personal registration information; however, a most common trait among this class was that contact exchange

of the user does not leave the user's device to preserve privacy and ensure the anonymity of interactions. However, there is a loss of efficiency if the user decides to not act upon the fact that they may be potentially infected raising the number of infections. There are many bandwidth costs associated with both servers to serve the data (egress costs) and to download the data (bandwidth costs) at the user. There is an associated battery cost, and the analysis of the contact exposure is very limited to low computational power.

- `Hybrid Architectures` As a compromise between centralised and decentralised architectures, Hybrid architectures were proposed. DESIRE[6] is an example of such where each encounter with a user is a unique token. In DESIRE, User regularly asks the server if they have been in contact with an infected user. The health care authorities do not need to bear the egress costs as the ingress costs are free in a single region and the user saves much computational power as they do not need to do risk analysis on their device. DESIRE expects new Ephemeral Bluetooth Identifiers (EBID) to be generated at each epoch to advertise, which may not be possible for a BLE as few specifications expect a certain delay between consecutive scanning or advertising intervals[5]. Linkage attacks are highly probable if new ephemeral ids generated are not random at all and at regular intervals.

We have decided to go ahead with a decentralised solution as it provided utmost privacy and security which is the objectives of this course. We have studied various decentralised solutions and how they differ to address some specific concerns like scalability and security attacks like replay/relay attacks.

2.1 Flavors of Decentralised Solutions

Most of the decentralised/hybrid approaches were successful against replay attacks, linkage attacks or prevention of social graph analysis. However, relay attacks were still a cause of concern. Pietrzak [22] proposed encoding of GPS co-ordinates and least significant of time as tags to ephID and stores them along with it. If the two contacts are indeed at the same location, the user can derive the same signature for the commit to producing the same tag. However, this may not be possible as location permission may not be available and the coarse location may be too broad that relay attacks are still possible.

Another approach to countering relay attacks was proposed based on Zero-Knowledge Proofs in Liu et al. [19]. In Liu et al. [19], Users compute and exchange a zero-knowledge proof of the contact, and on positive diagnosis, they upload the zkp-proofs to the central server. Group signatures at the health authority level were proposed to protect the identity of the confirmed patient. Each user downloads last 14 days keys and checks if they have been in contact with a user. However, this approach is not scalable because the user is not uploading their respective Temporary Exposure Keys (TEK), but rather, all the connections they have had with other users. Storage at the server end can increase exponentially. The authors hence suggested this approach at the local level, like a city or a state. However, A single city or a state may not have the resources to do it. On the device level, the author expects the devices to be paired using Bluetooth, which may be energy-consuming and contacts miss (true-negatives) even if false positives are significantly less.

Decentralized solutions have received widespread acceptance over the last few months, Decentralized Privacy-Preserving Proximity Tracing (DP-3T)[24] and GAEN[13] being the prominent ones. The bandwidth costs and risk score analysis needs refinement to improve accuracy and reduce false-positive cases. GAEN has published Bluetooth Low-Energy (BLE) Attenuation figures to correctly estimate the distance based on Received Signal Strength Indicator (RSSI) and encrypted txPower in the payload. Vukolic [26] tries to address the bandwidth costs of a decentralised solution by subdividing regions into base and roaming regions and syncing the region servers with all infected TEK. The user downloads from both the base and travelled regions TEK for risk analysis. Vukolic [26] also addresses interoperability between various platforms by using a common BLE payload and does not assume that there are strict social distancing measures in place to forbid inter-state travelling like in Liu et al. [19].

Some of the concerns we had were more legal than technical in nature because contact tracing happens every minute and deals with the health data of the user. HIPAA violations cost a lot of money and any contact tracing application has to be legally compliant. Bradford et al. [7] provided few guidelines and some fundamental ideas. For instance, Is it legal for the state to ask for data as the powers of the state can persuade the user to share it forcefully. It states that pseudo-anonymous should not determine the health status of the person. It is important to fetch the health status using a Transaction Authorization Number (TAN) which is not reusable to be HIPAA-Compliant. The paper provided us guidelines on what responsibilities we might have as Data-Controller and guarantees that we need to provide. It helped us choose our cloud provider for the project.

3 COVIDGUARD-F APPLICATION

Our application is based on GAEN architecture with a few minor deviations. For the server-side architecture, We have taken references from Covid-Warn App [3], which divided the concerns of each authority into three different entities. In the following sections, We would like to discuss a brief overview of the approach we have taken and my personal contributions. We have not been given access to GAEN API from Google/Apple as the Exposure Notification API is only available for Sovereign Entities. We have developed our solution following the cryptographic specification[1]

3.1 Current Approach

We have deviated from the GAEN in some aspects to simplify the development and meet the privacy and security criteria. We advertise the BLE at regular intervals of one minute intervals and change TEK every five minutes. We do not change the RPI when the Bluetooth MAC address changes. To make sure our implementation is cryptographically secure, We derive our Rolling Proximity

[1]https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-CryptographySpecificationv1.2.pdf

Identifier (RPI) using the following equation.

$$\begin{gathered} RPI - KEY = PRF(PRG(Seed)), \\ RPI = AES128(RPI - Key, PaddedData) \end{gathered} \tag{1}$$

We are using the Java Secure Random with SHA1PRNG as the Pseudo-Random Number Generator (PRNG). To make sure all the outputs from the PRNG appear random, We are using HMAC-based Extract-and-Expand Key Derivation Function (HKDF)[16], A Pseudo-Random Function Family (PRF) to extract a key for generating RPI. If the user is tested positive, We upload the keys to the Exposure Notification Service (ENS) server. The user registers with the health care using a QR code generated and written with the verification server. The verification server checks with the health care server if the user is indeed positively diagnosed. The app periodically downloads the Infected User's TEKs and regenerates RPI to match with the RPIs received. If there is a match, We notify the user to get tested.

3.2 Server-Side Work

On the server-side, We have divided the interactions of the user into several components to address legal compliance issues and privacy concerns. ENS Servers need not know the user-related information and Verification servers should not see the health care details of their user. Health care server has to be HIPAA Compliant to interface with Diagnostics Centres. We have separated the concerns, as shown in Figure 1.

3.2.1 Health Care Server. The Health Care server may be the only server that may contain personal details about the user, and it requires the diagnostic centres to upload test results for a particular user. To address some of the privacy concerns and give users control over their data. We have added a consent layer and QR code-based mechanism for the diagnostic centres to upload test results with UUID tagged belonging to the user. Our Platform's Health Care Servers can always query the health care authorities regarding the test results of the user.

Verification server checks with the health care authorities if the user requesting TAN is indeed tested positive. User has to give consent to the verification user to fetch their details. If the user is tested positive, Verification server will generate a TAN to upload their TEKs. If the user revokes the consent, Verification server cannot check the health status of the user. I have provided the API references in Appendix **??**. To mock the health care authorities interaction, I have defined API to register a user and set their status.

3.2.2 Verification Server. Verification Server is the main API that interacts with the app and provided server-side functionality for its features. It is the owner of user application data and is tied to a single app installation. If the user uninstalls the app, loses the UUID and Token granted. Any fresh install will treat the user as a new user and generate a new token. In this way, We maintain the pseudo-anonymity of the user. User does not interact with the health care server outside the purview of a diagnostic centre and registration with the corresponding authority. Health Care servers host sensitive data and act as an interface between health authorities and the citizens. Verification servers serve as an intermediary for checking health care status if the user has agreed to the terms & conditions and provided consent.

The user registers the UUID with the verification server, and the verification server grants a token for subsequent communication. For the project, We have only provided the TAN functionality. When the User requests a TAN, TAN is provided if the User is indeed tested positive, User uploads the TEKs along with the TAN provided to the ENS server, ENS Server checks the TAN using Verify-TAN API. If the verification server verifies the TAN as valid, It will authorise the upload of TEKs. Separation of concerns between Verification Server and ENS server allows us to accommodate different health care providers by providing a common interface to implement for all of them.

3.2.3 Exposure Notification Service. The Exposure Notification Service is only concerned with upload and download of infected keys. We run a cron job to clean up keys older than 14 days every day. The ENS server verifies TAN with verification server and allows the upload of TEKs. Presently, We do not expose all the keys by aggregation as a static file to download. Instead, We make it a POST request to download the keys periodically given a small size. For a production setup, We propose to periodically aggregate the Diagnosis keys using a batch data processor and exposed via Content Delivery Network (CDN).

3.2.4 Separation of Dev and Production for fast debugging. To make sure the development of Server APIs does not block the development in client-side functionality. I needed a staging environment. However, Firestore only allows a single database per project, and we do not want to maintain separate projects for each environment of a server as that might not scale for our use case. It may cost a tiny bit even though all the services we are using are free. FireStore does not have an emulator for local testing, so we had to create namespaces in the production cluster itself.

FireO[2] - Firestore Object–relational mapper (ORM) library was very useful to abstract the semantics away and provided a class-based view for our data models. In the metadata, If we are running a server in the local environment or development environment. We added a namespace 'dev_' to the collection name in our models. Local testing did not have an impact on client-side implementation as the data was separated between environments. Android Application only interacted with the production server setup, which was stable after testing in the developer environment.

3.2.5 Reasons behind the choice of infrastructure. We have used Google App Engine (GAE)[3] for hosting the servers as it provided us monitoring abilities and auto-scaling functionalities as it is a serverless platform. Google addressed concerns on instance security and SSL security. It has support for continuous deployment scenarios like traffic splitting and blue-green deployments, which helped us to update our server code as a constant process after the app is released. If an erroneous deployment takes place, We can immediately split the traffic to mitigate the errors. GAE also provided us with Audit Logs in compliance with Australian Privacy Principles (APP) and Access Control Mechanisms to manage the staff for

[2]https://octabyte.io/FireO/

[3]https://cloud.google.com/appengine

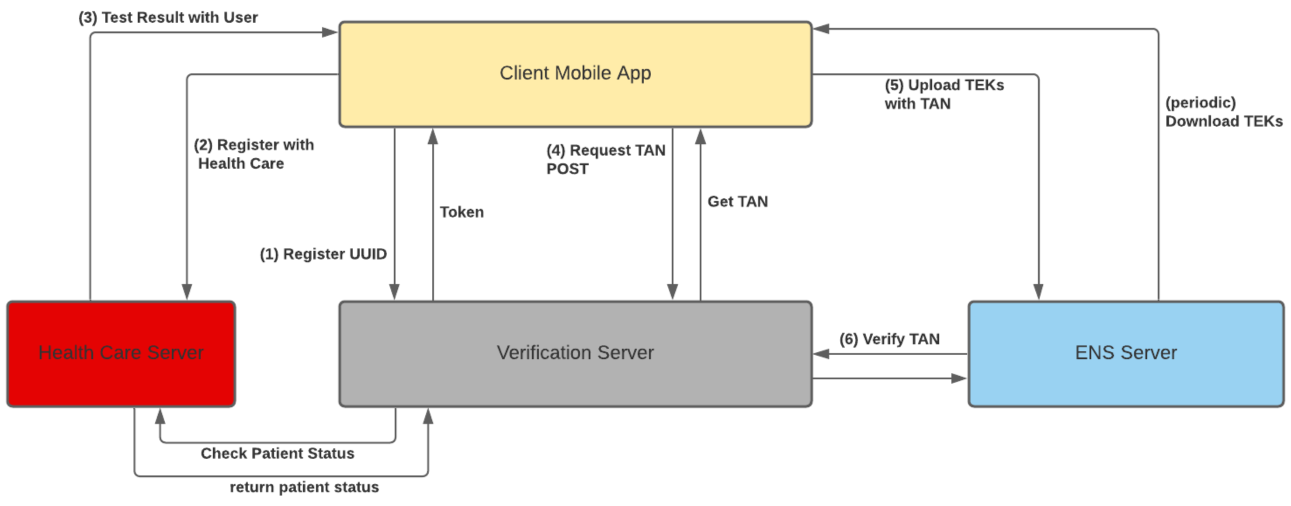


Figure 1: Backend Architecture of CovidGuard-F

free. It provided us with servers in Australia's Jurisdiction and also provided us with some standard libraries. I am confident that it is indeed secure to host an exposure notification platform.

Google Cloud provided us with advanced authentication mechanisms like password-less authentication to work with the infrastructure and had support for terraform[4] - Infrastructure-as-code (IAC) software. We can develop and deploy software securely even if the source code is exposed to the public, which can provide much transparency to the public on what we do in the cloud.

3.3 Client-Side Work

In this section, We would like to describe the work we have done on the android application to address functional requirements and make the application as responsive as possible. We have tried to use only popular open-source repositories or official android libraries to avoid security vulnerabilities and leverage open-source support for auditing and servicing of the dependencies.

3.3.1 ROOM-Based Storage. In the client-side, We have first done direct SQLite based database setup. However, it was not very straightforward with multiple tables. My teammate had to create a database per table to store generated TEKs and received RPIs. We could not integrate SQLCipher[5] to protect data at rest that easily. We had to write raw SQL queries based on the app inputs which were exposed to SQL Injection attacks.

Room supported reactive applications by providing a direct interface with LiveData and RxJava. It allowed for much extensibility with interfaces to write clean code that can interact with SQLite database. When we try to add new stakeholders to the app and give access to multiple healthcare servers, we can easily extend the database by using migrations which is not very straightforward in native SQLite libraries. We have described the schema used in the Application in Appendix **??**.

3.3.2 Networking using Retrofit. We initially used moshi[6] and OkHTTP[7] for writing the executors. OkHttp based network calls required much boilerplate code, and it was very tiring to group multiple API calls into a single logical service. I wanted to use grouping to define all APIs of a particular server as a service. This allowed me to define authentications schemes for each server. As the number of servers increased to three and the number of API calls increased in multiple, It was harder to define each request and response types.

Retrofit[8] allowed me to define API Calls as Annotations and the service as Java Interface. It has also allowed us to determine how we interact with the service calls and implement SQLPinning effortlessly. Using Work Manager, We were able to chain multiple API calls to get our application working.

3.3.3 Background Processing using WorkManager. Work Manager is an androidX jet pack library for background processing of jobs. In CovidGuard, There are few tasks like the periodic download of TEKs and matching them with received RPIs. WorkManager allows us to make DAG graph of tasks chained together. If a parent task fails, child tasks do not get executed. Work Manager also allowed us to chain tasks together like requesting TAN and uploading Diagnosis Keys which are two different tasks needed to submit the TEKs. The tasks I have defined in the android app are -

- `Check Patient Status` Checks the patient status registered with the UUID in the healthcare server.
- `Match Maker Task` Downloads the infected user's TEKs and matches with the RPIs. After my teammate has defined the code, I have wrapped it under a work manager task.

[4]https://www.terraform.io/
[5]https://www.zetetic.net/sqlcipher/
[6]https://github.com/square/moshi
[7]https://square.github.io/okhttp/
[8]https://square.github.io/retrofit/

- `Request TAN Task` Requests a TAN for uploading TEKs.
- `Upload TEKs Task` Upload TEKs to the diagnostic server.

Request TAN Task and Upload TEKs Task are chained together and if the Request TAN Task fails, Upload TEKs task is also not executed. Network requests and Disk operations can be performed synchronously as work manager does not run on the UI thread. This allowed a lot of flexibility in writing the algorithms.

3.3.4 Refactor of RPI Generation. In the earlier version developed by my teammate, Two tasks were scheduled as per TEK Interval and RPI Interval set in the code. We had not yet developed a database layer. There were a couple of issues with this approach. Few of them were -

- **Dangling RPIs and Race Conditions** Scheduled executors are not precise in nature. There might be a delay of a few seconds that can add up for subsequent operations. There is a scenario Advertised RPI is based on the previous TEK, and the EN Interval Number (ENIN) is above the TEK Rolling Period. In such cases, We cannot match the RPI if the tolerance window is not present for generating RPIs.
- **Too many advertisements** Advertisements launched during earlier RPI cycles were still being advertised, and as the app worked for more extended periods, new promotions failed.
- **Shared Volatile Variables** The RPI scheduler service generated TEK stored as a shared variable which is edited by a scheduled executor service. If the TEK variable being a shared variable is edited again in the Activity class, RPI Scheduler will keep failing.

To address these concerns, I have devised an algorithm to generate RPI by fetching TEK from storage -

Algorithm 1 New RPI Generation

```
procedure getRPIToAdvertise(-)
    $TEK, ENIN \leftarrow getLatestTEKFromDatabase()$
    $PRESENT\_ENIN \leftarrow calculatePresentENIN()$
    if $PRESENT\_ENIN - ENIN \geq 5$ then
        $NEW\_TEK \leftarrow generateNewTEK()$;
        $StoreNewTEKInDatabase(NEW\_TEK, PRESENT\_ENIN)$
        $TEK \leftarrow NEW\_TEK$
    end if
    $NEW\_RPI \leftarrow generateRPI(TEK, PRESENT\_ENIN)$
end procedure
```

Algorithm 1 allowed us to strictly bound an RPI to a five-minute interval of a TEK. If a TEK is not generated within the last five minutes, a new TEK is generated. New TEK generations ensure that we are not missing matching of any RPI when the TEKs are shared with ENS server.

3.3.5 Consent for accessing location. In Android, We need the location permission to access the BLE functionality. Android categorises the permissions as sensitive and normal permissions, internet and background state are standard permissions which do not require explicit consent from the user. Sensitive permissions which can compromise user privacy or have the potential to mine user data need explicit consent from the user. In adherence to APP[1], We have solicited the information explicitly and if the user denied it, We have provided explanation on we need the permission. If user explicitly denies it, We show them what functionalities, may not be available. In Figure 2, We have illustrated the flow the user will go through if they refuse consent and how they redirected by their choices.

3.4 Concerns Addressed By Me

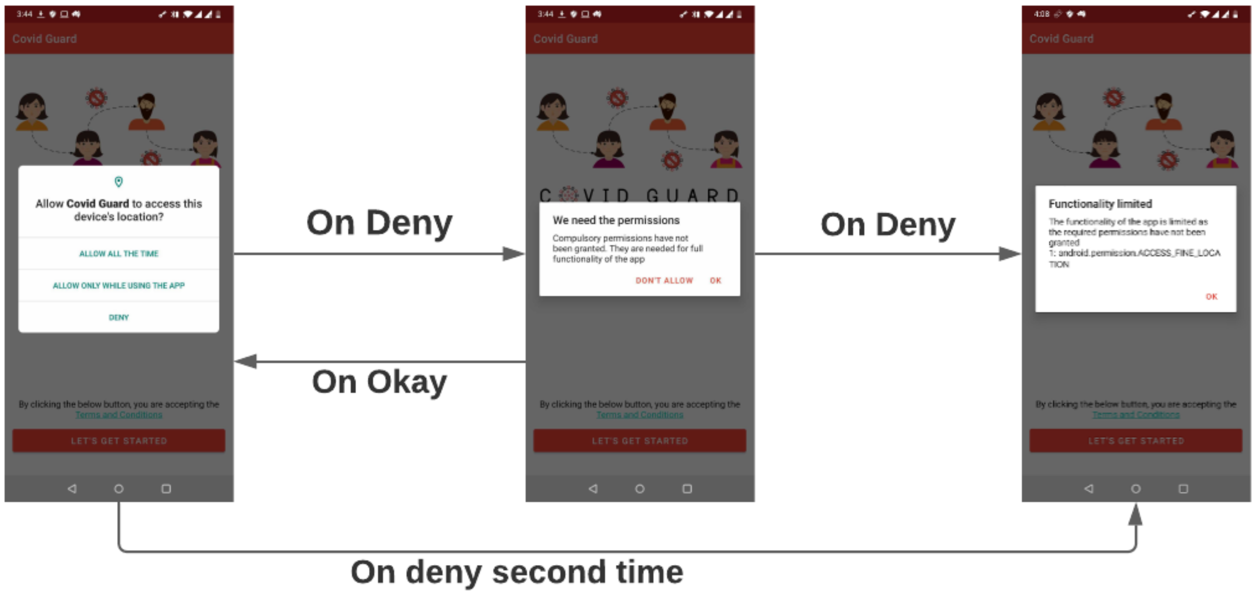
3.4.1 Encryption at Rest. We have encrypted the token shared by verification server and UUID associated with it in encrypted shared preferences offered by android libraries. They cannot be accessed even if the developer has access to debugging tools to check the data stored by the application. The keys in the encrypted shared preferences are encrypted deterministically using AES256-SIV-CMAC[14] and values using AES-GCM. AES-SIV-CMAC is a deterministic scheme that does not compromise on the key security even if the nonce is not provided. The attacker will know that the same key is being used but not the key itself. In case of a map, keys are distinct. Hence, the scheme is suitable for encrypting key names in shared preferences.

We have also encrypted the database using SQLCipher[9] free version. The passphrase for the database is generated when the database is created and persists with the app installation. The passphrase is unique to every time the app is installed. SQLCipher uses AES encryption to encrypt the database.

3.4.2 Encryption in Transit. Encryption in transit is guaranteed by the TLS 1.3 protocol provided by Conscrypt[10] in older android versions. In the new android versions, TLS 1.3 is enabled by default. TLS 1.3 deprecates older cipher suites and provides perfect forward secrecy that ensures session keys are not compromised if the generating keys are compromised. This is very useful to prevent Man-in-the-middle (MITM) attacks; however, it does not prevent censorship. Using our application, organisations can run very secure and privacy-sensitive contact tracing solution, which is independent of an authoritarian government. The adversarial government can still ban the traffic to the servers as server name indicators are not encrypted in an HTTPS connection. TLS 1.3 has support for Encrypted Server-Name Identifier (ESNI) to prevent blocking of HTTPS traffic by SNI header and prevent censorship by the server.

To boost the TLS-based security, We have added SSL pinning - hard-coded SHA-256 certificate hashes to only allow communication if the certificate tree has a particular certificate. In our case, since google supports APP compliance and provides tools to be compliant, We restricted the access usage to google organisation-wide certificate only. We initially restricted it to Google App Engine URLs and later realised we might need to interact with other services as well. A significant drawback of this approach is that the user has to update the mobile application periodically to get the new certificate hashes.

[9]https://www.zetetic.net/sqlcipher/
[10]https://www.conscrypt.org/



Allow **Covid Guard** to access this device's location?

We need the permissions

Compulsory permissions have not been granted. They are needed for full functionality of the app

Functionality limited

The functionality of the app is limited as the required permissions have not been granted
1: android.permission.ACCESS_FINE_LOCATION

By clicking the below button, you are accepting the Terms and Conditions

Figure 2: Permission Management flow

3.4.3 Background Processing of Jobs. Android has deprecated many async APIs and asked developers to use java concurrency libraries. Android does not allow Network and Disk operations on the main UI thread and requires developers to utilise asynchronous functionality. A lot of literature on how to do it depended on AsyncTask provided by android which has recently been deprecated and we had to work on a new approach. One of the options was using RxJava[11], which leverages the observer pattern to post-process responses from DB or servers. However, RxJava has a steep learning curve, and we felt it was unnecessary for a project of small magnitude.

Android provided LiveData as an observer data holder class to update data in activities, fragments and lifecycle-services. However, Libraries also have to provide support for it. Android Room and Retrofit provide support for livedata; nevertheless, this is not extensible. So we have defined a Singleton class holder with several threads for network IO, DiskIO and ScheduledIO. This allowed us to process asynchronously and get futures to listen on. However, ListenableFutures are returned only by ListeningExecutorService, which is available in android Guava library, which is very massive. So we have used a combination of measures for streamlining asynchronous listeners and processors.

We have used retrofit API to replace as many calls to networkIO as possible and retrofit provided callbacks to handle post-processing of responses. Android Room provided LiveData holders, and we used diskIO thread if it was not possible. However, WorkManager[12] later solved most of our problems as it supported constraint-based scheduling of tasks, backoffs and retries of tasks, chaining of different tasks and listening to results from WorkManger tasks. It would help the breaking of task chains if a parent task failed. It had excellent support for Periodic Schedulers (up to 15 minutes precision) and handled edge cases on what to do if a periodic job from earlier is still running.

We have chained requesting TAN and upload of TEKs - two different tasks using work manager. We have defined constraints to match infected TEKs based RPIs with received RPIs only when the battery is not low, and network connection is available. We have achieved significant gains in performance using WorkManager. We could not wholly deprecate AppExecutors Singleton because RPIGeneration has to happen in a single-minute interval and it is not a background service. ScheduledThreadService and observers solved the issue of advertising in single minute intervals.

3.4.4 Separation of concerns at Server and Auditable Infrastructure. We have defined our entire infrastructure using Terraform and used Google Service Account Impersonation[13] to provide access to teammates without sharing credentials and passwords. We have used a remote terraform state stored in an S3 bucket and encrypted using Google Managed Encryption Keys so that changes made by a single teammate is visible to all instantaneously. We have followed a shared-nothing architecture and used microservices paradigms for inter-service communication. Each app engine has access to only its individual firestore database, as shown in Figure 3. We have separated the collections in the database by namespacing the collection names; developer environment appends *dev_* to the

[11]http://reactivex.io/
[12]https://developer.android.com/topic/libraries/architecture/workmanager
[13]https://cloud.google.com/iam/docs/impersonating-service-accounts

collection names to separate collections from the production environment.

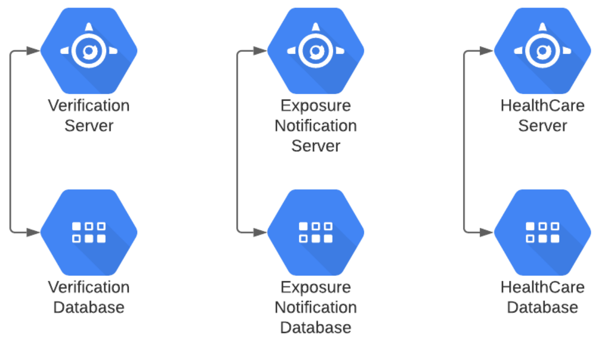


Figure 3: Google App Engine and Google Firestore backends

3.4.5 Resolving failed advertisements. In the earlier architecture of TEK and RPI generation, BLE Advertisement was done inline using shared variables to access RPI. Previous RPIs were not stopped before a new RPI was generated. This had the potential to create bugs in the matching phase as the RPIs received were outside the tolerance window of a TEK.

In the new architecture, I have implemented the BLE advertiser as an observer to RPI generation service. I have made the BLE Transmitter a static variable and advertising wrapper function as a synchronised function to prevent race conditions. The advertiser is notified on RPI Generation, and it stops the previous advertisement and starts the advertisement for a new RPI.

3.5 Assessments Done by Me

3.5.1 Load Testing. A decentralised architecture has a high egress cost and bandwidth costs for a user because the user has to download the keys periodically. The infected diagnosis keys are exposed via an endpoint. Requests might fail at large scale, and we are not using cache anywhere to avoid queries to the database. I have done load testing to make sure we can handle the load of at-least 1000 users. We have documented the results in Table 1.

Request Rate	100 requests / sec
Duration	10s
p99 Latency	6.57s
p95 Latency	5.67s
Bytes in Total	2.2 MB
200 Status	1000
503 Status	0

Table 1: Load Testing Configuration and Results

3.5.2 Server Costs. We have used google cloud cost billing and incurred low costs. We were within the free tier limits of the Google Cloud Engine. Our projected total costs were 0.12] USD per month for each project. We have only incurred costs for storing terraform state and encrypting them using Google Key Management Service (KMS) service. App Engine provides 28 instance hours free, and we have enabled shutdown instances if they have not received requests instead of always on. App Engine being serverless allowed us to incur a smaller penalty on restart time and warm-up requests to make sure instances are healthy to serve when a new model is created. Google Firestore provided 50,000 reads and writes free with 5GB storage per instance. '

3.5.3 Web Server Vulnerability Scanning. I have run Nikto[14] webserver scanning tool and hosted the results in our GitHub. Nikto has not any severe vulnerabilities in our server and nikto scan made around 8000 requests scanning for cms-plugins, sitemaps, CGI scripts and WordPress bugs. Nikto did highlight the use of wildcard certificates. We essentially would like to provide unique credentials to each of our servers.

3.5.4 Adherence to Australian Privacy Principles. We have provided auditable infrastructure and addressed the concerns raised in Vaudenay and Vuagnoux [25] regarding the latest snapshots of code that are providing service to the users. We have not addressed deficiencies in GAEN Protocol. However, We have provided full-scope of client, server and infra as code without any secrets to the public for auditing. We have also offered audit logs of all our servers and only associated personal information with healthcare API, which is subjected to Australian Privacy Principles that provide us with a framework on how to handle personal data of users. We have added consent API to auto-delete or deny information if the consent is revoked.

By making our code public, We are adhering to APP principles - 1 and 2 [1]. Scope of personal information is limited to HealthCare API concerning APP 3. We have provided terms and conditions, and explained user using permission workflow on why we need the location permission. Verification server interacts with the healthcare server using pseudo-anonymous identifiers, and hence personal information is never disclosed unless the user provides the consent. This is in accordance with APP 5 and APP 6. We have not added any direct marketing platforms hence compliant with APP 7. We have hosted all information in the Australian region and do not share information with any cross border entities.

We have not provided HealthCare dashboards for users to interact with their personal information or personal encryption keys. We are hence not compliant with APP 10-13. However, We need not make any significant changes to our architecture to be compliant with other APP principles.

4 FUTURE WORK

4.1 Risk Score Analysis

We are doing a naive-matching of received RPIs with the ones generated from downloaded TEKs. There is scope for M-of-N Detectors suggested in Hatke et al. [15] to adjust according to the RSSI values of the received RPIs. This also addresses concerns raised on authentication of encrypted metadata transmitted in GAEN protocol [25]. We can also adjust BLE power as per device to get more accurate

[14]https://cirt.net/Nikto2

distance measurements. We do it by changing the transmission power of the BLE. Google has provided a list of attenuation levels for a lot of devices to get accurate distance readings. Based on the distance and number of matched RPIs, We can classify the user as Low, Medium or High risk to COVID-19 infection. United Kingdom National Health Service (NHS) has devised a Risk-Scoring algorithm for measuring distance every 3.6 minutes and classifies the user as close contact, or low-risk user [8]. Estimation of distance from BLE transmission is an active area of research, and a few other methods include a measure of distance by identifying advertising channel[12].

4.2 CDN based download of TEKs

We are providing a REST-Endpoint to download the TEKs; there is a need for caching of responses and low latency delivery to the user. We have to pre-process the infected diagnosis keys, aggregate them and serve as a single file download with a SHA256 hash of the data to the user for integrity checks. This required money and elaborate data-processing setup which fetches data for the last 14 days. We would like ideally to approach this using a single machine beefy machine. In case of a large number of users, We may have to divide the file in chunks so that user may download keys sorted by an hour or minute. The granularity of the file may depend on the size of the data. This approach has some drawbacks like user needs to know the number of files they have to download.

Another approach is to provide a streaming API with limits and offsets, and the user can make multiple calls to the server to fetch the next chunk. Server-side caching will reduce the load on DB, and the user can download smaller files to reduce the rate of failure due to large files in case of erratic internet connections in developing countries. Vukolic [26] suggested dividing the regions to reduce the size of the infected TEKs per region and also advocated for a standard payload format for interoperability between different contact tracing applications.

Negotiation with a cloud provider for better pricing on egress costs can reduce the cost of decentralised contact tracing. We incur egress costs to facilitate the download of infected TEKs for each user of the app. Recently, Zoom Inc has chosen Oracle Inc for data-centre hosting to cut down on egress costs[15]. If we are selecting a single-file approach, We can also host the file on a CDN to facilitate low-latency download across the globe.

4.3 Client IP Address and Adherence to APP-4

One of the concerns we have not addressed in our application is related to APP-4, dealing with unsolicited personal information. Cloud Provider and others log the client IP-address when they make the request to servers. Client IP Address can be used to identify the user. Lindell [17] suggested that the request should be passed through a mix of routing servers before reaching the ENS servers to protect the user identity.

[15]https://www.lastweekinaws.com/blog/why-zoom-chose-oracle-cloud-over-aws-and-maybe-you-should-too/

Client IP Addresses have to be anonymised before sharing them with third-party services for intrusion detection and verification of audit logs. At scale, there are multiple services which generate these logs and stored in a central logging repository. Xu et al. [28] introduced a parallel version of TCPDPriv[20] with similar security guarantees that do not depend on the order of the logs. For a PRF and a chosen key, original IP Address is always mapped to an anonymous address; hence, records can be divided in chunks for parallel anonymisation process.

Eckert and Pircher [11] suggested that we delete identifiable data before logging it in our servers. For instance, Referrer, X-Forwarded-For, Cache-Control and User-Agent headers have to be scrapped by the server or gateway service before passing it to ENS servers. However, we need some of the data for rate-limiting of requests to prevent Denial-Of-Service (DOS) attacks. Anonymisation of logs is a middle-ground to support both functionality and compliance efforts.

4.4 Authentication and Audit Trails of HealthCare

Presently, We do not have any authentication between health care servers, users and verification servers. Healthcare servers hold personally identifiable information of the users and their health information. HIPAA audit trails require four of the following details in their audit trails as described in [18] -

- Authentication information of the staff who accessed the data
- What data has been accessed?
- On which servers or who catered the data to users?
- Timestamp of when the data was accessed by the user.

Substantial investment in auditing infrastructure is needed and to support it, We have presently defined logging as dictionary configuration and separated the formatters and handlers. This will allow us to define a custom handler for audit trails and do filtering by tag. We can add 2-Factor authentication and SSO-based login for healthcare staff and active directory service for authorisation purposes. Liu et al. [18] suggested that we divide the audit trails architecture into the notification, record layer, audit layer and action layer. Action layer's duties may include locking the patient details in the case of unauthorised access. Notification layer is for authorities to know in real-time that this data has been accessed. Audit Layer deals with the authentication of the user accessing the data. Record layer contains information on the changes in data and authorisation lists on who can access the data.

HIPAA also provided some guidance regarding the physical security of the system. We have addressed the employee security by Google Service Account Impersonation of a limited developer role. We can add further restrictions of 2-Factor authentication and rotation of passwords. There are some other tasks that are left like auditing of third-party libraries, pinning third-party libraries using SHA256 hashes, and using google secrets manager[16] for storing server-side

[16]https://cloud.google.com/secret-manager

credentials. However, there are minor concerns and can be quickly addressed.

4.5 Exploring Hybrid Architecture

Good hybrid architectures work with pseudo-anonymous data that is uploaded to the server to protect user privacy. Hybrid architectures may appear cheaper than decentralised solutions and use less bandwidth because the risk analysis is done on the server-side. However, the cloud economics of hybrid architectures depends on the size of the disk usage if all the RPIs have to be stored vs the cache egress costs.

Google published that network costs are above 10,000 USD for 1250 daily active users if 84TB has to be downloaded monthly[17]. If we store all the RPIs generated for all the users, Storage costs maybe 5x the storage cost of keeping just TEKs and compute costs are already cheaper if the request need not be sent responded to immediately. Hybrid architectures also take less battery consumption on the user side as risk analysis is a computationally heavy task. This may encourage more users to download the application.

5 CONCLUSION

Decentralised contact tracing sacrifice time-taken to act on potential positive infections in favour of the privacy of users. Contact tracing solutions can be used for infectious reasons if the centralised server is not honest-but-curious servers and actively track the users. GAEN protocol addresses a lot of concerns on privacy, but it is limited to use by only sovereign entities for COVID-19 infections. A privacy-sensitive contact tracing solution can be used for other purposes and disease tracking. Few other applications include tracing of customer movements in a shopping centre or movement of personnel in a hospital for building better structures. We have provided a custom GAEN-based contact tracing platform and separated the parts which deal with personal information for privacy-conscious scenarios. We have provided code that actually runs in the server in a secure way without exposing credentials to offer transparency in our solutions and leverage open source community to help us provide a better solution.

There is a lot of work left in making sure we are compliant with various legislations, APP and GDPR in particular. We have to set up an audit logging infrastructure and support for 2-factor authentication of staff. We need to negotiate with cloud services which do not charge retail prices on egress data for hosting this solution to provide a low-cost alternative to popular GAEN protocols. Further research is needed if peer-to-peer protocols like BitTorrent can be used to share infected user TEKs, accurate risk analysis of contacts and measurements on differences in BLE strength of popular devices. Decentralised solutions have to tackle a time lag between confirmation of infection and notification of possible contact. If the periodic downloads

REFERENCES

[1] 1988. The Privacy Act 1988. https://www.legislation.gov.au/Series/C2004A03712

[2] 2016 (Accessed 20 Oct 2020). *European Parliament and Council of European Union (2016) Regulation (EU) 2016/679.* https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN

[3] 2020. *Corona-Warn-App Open Source Project.* https://github.com/corona-warn-app

[4] N. Ahmed, Regio A. Michelin, Wanli Xue, S. Ruj, R. Malaney, Salil S. Kanhere, A. Seneviratne, W. Hu, H. Janicke, and Sanjay Jha. 2020. A Survey of COVID-19 Contact Tracing Apps. *IEEE Access* 8 (2020), 134577–134601.

[5] AltBeacon - The Open and Interoperable Proximity Beacon Specification 2020 (Accessed 20 Oct 2020). *Beacon Manager JavaDoc.* AltBeacon - The Open and Interoperable Proximity Beacon Specification. https://altbeacon.github.io/android-beacon-library/javadoc/reference/org/altbeacon/beacon/BeaconManager.html#getForegroundBetweenScanPeriod()

[6] Nataliia Bielova, A. Boutet, C. Castelluccia, M. Cunche, Cédric Lauradoux, D. Métayer, and V. Roca. 2020. DESIRE: A Third Way for a European Exposure Notification System (SUMMARY - EN).

[7] L. Bradford, M. Aboy, and K. Liddell. 2020. COVID-19 contact tracing apps: a stress test for privacy, the GDPR, and data protection regimes. *Journal of Law and the Biosciences* 7 (2020).

[8] M. Briers, Marcos Charalambides, and Chris Holmes. 2020. Risk scoring calculation for the current NHSx contact tracing app. *ArXiv* abs/2005.11057 (2020).

[9] Centers for Medicare & Medicaid Services. 1996. The Health Insurance Portability and Accountability Act of 1996 (HIPAA). Online at http://www.cms.hhs.gov/hipaa/.

[10] Lisa O. Danquah, Nadia Hasham, Matthew MacFarlane, Fatu E. Conteh, Fatoma Momoh, A. Tedesco, Amara Jambai, D. Ross, and H. Weiss. 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 (2019).

[11] C. Eckert and A. Pircher. 2001. Internet Anonymity: Problems and Solutions. In *SEC*.

[12] C. Gentner, D. Günther, and Philipp H. Kindt. 2020. Identifying the BLE Advertising Channel for Reliable Distance Estimation on Smartphones. *ArXiv* abs/2006.09099 (2020).

[13] Google and Apple 2020 (Accessed 20 Oct 2020). *Privacy-Preserving Contact Tracing - Apple And Google.* Google and Apple. https://covid19.apple.com/contacttracing. Accessed20Oct2020

[14] D. Harkins. 2008. Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES). *RFC* 5297 (2008), 1–26.

[15] Gary F. Hatke, M. Montanari, S. Appadwedula, M. Wentz, J. Meklenburg, L. Ivers, J. Watson, and P. Fiore. 2020. Using Bluetooth Low Energy (BLE) Signal Strength Estimation to Facilitate Contact Tracing for COVID-19. *arXiv: Signal Processing* (2020).

[16] H. Krawczyk and P. Eronen. 2010. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). *RFC* 5869 (2010), 1–14.

[17] Yehuda Lindell. 2011. Anonymous authentication. *Journal of Privacy and Confidentiality* 2, 2 (2011).

[18] B. Liu, Zheng Zhou, and H. Huang. 2005. A HIPAA-Compliant Architecture for Securing Clinical Images. *Journal of Digital Imaging* 19 (2005), 172–180.

[19] J. K. Liu, M. Au, T. Yuen, Cong Zuo, Jiawei Wang, A. Sakzad, X. Luo, and Li Li. 2020. Privacy-Preserving COVID-19 Contact Tracing App: A Zero-Knowledge Proof Approach. *IACR Cryptol. ePrint Arch.* 2020 (2020), 528.

[20] G. Minshall. 1997. Tcpdpriv: program for eliminating confidential information from traces.

[21] Romualdo Pastor-Satorras and Alessandro Vespignani. 2001. Epidemic spreading in scale-free networks. *Physical review letters* 86 14 (2001), 3200–3.

[22] K. Pietrzak. 2020. Delayed Authentication: Preventing Replay and Relay Attacks in Private Contact Tracing. *IACR Cryptol. ePrint Arch.* 2020 (2020), 418.

[23] K. C. Swanson, Chiara Altare, C. S. Wesseh, Tolbert G. Nyenswah, Tashrik Ahmed, N. Eyal, E. Hamblion, J. Lessler, D. Peters, and M. Altmann. 2018. Contact tracing performance during the Ebola epidemic in Liberia, 2014-2015. *PLoS Neglected Tropical Diseases* 12 (2018).

[24] C. Troncoso, M. Payer, J. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, S. Chatel, K. Paterson, Srdjan vCapkun, David Basin, J. Beutel, D. Jackson, Marc Roeschlin, P. Leu, B. Preneel, N. Smart, Aysajan Abidin, Seda Gurses, M. Veale, C. Cremers, M. Backes, Nils Ole Tippenhauer, Reuben Binns, C. Cattuto, A. Barrat, Dario Fiore, M. Barbosa, R. Oliveira, and J. Pereira. 2020. Decentralized Privacy-Preserving Proximity Tracing. *ArXiv* abs/2005.12273 (2020).

[25] Serge Vaudenay and Martin Vuagnoux. 2020. Analysis of SwissCovid. (2020). http://infoscience.epfl.ch/record/278048 This report was submitted on 5.6.2020 to https://www.melani.admin.ch/melani/en/home/public-security-test/infos.html.

[26] M. Vukolic. 2020. On the Interoperability of Decentralized Exposure Notification Systems. *ArXiv* abs/2006.13087 (2020).

[27] World Health Organization 2020. *Coronavirus Disease 2019 (covid-19) situation report - 73.* World Health Organization. https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200402-sitrep-73-covid-19.pdf

[17] https://google.github.io/exposure-notifications-server/getting-started/estimating-costs.html

[28] J. Xu, J. Fan, M. Ammar, and S. Moon. 2002. Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme. *10th IEEE International Conference on Network Protocols, 2002. Proceedings.* (2002), 280–289.

[29] P. Zizler and Mandana Sobhanzadeh. 2020. Flattening the Curve.

ACRONYMS

APP Australian Privacy Principles. 3, 5, 7–9

BLE Bluetooth Low-Energy. 2, 5, 7–9

CDN Content Delivery Network. 3, 8

COVID-19 Coronavirus Disease of 2019. 1, 8, 9

DOS Denial-Of-Service. 8

DP-3T Decentralized Privacy-Preserving Proximity Tracing. 2

EBID Ephemeral Bluetooth Identifiers. 2

ENIN EN Interval Number. 5

ENS Exposure Notification Service. 3, 5, 8

ESNI Encrypted Server-Name Identifier. 5

GAE Google App Engine. 3

GAEN Google Apple Exposure Notification. 1, 2, 7, 9

GDPR General Data Protection Regulation. 1, 9

HIPAA Health Insurance Portability and Accountability Act. 1–3, 8

HKDF HMAC-based Extract-and-Expand Key Derivation Function. 3

IAC Infrastructure-as-code. 4

KMS Key Management Service. 7

MITM Man-in-the-middle. 5

NHS United Kingdom National Health Service. 8

ORM Object–relational mapper. 3

PRF Pseudo-Random Function Family. 3, 8

PRNG Pseudo-Random Number Generator. 3

RPI Rolling Proximity Identifier. 2–9

RSSI Received Signal Strength Indicator. 2, 7

SEIR Susceptible, Exposed, Infectious, Recovered. 1

TAN Transaction Authorization Number. 2–6

TEK Temporary Exposure Keys. 2–9