CONTAIN: Privacy-oriented Contact Tracing Protocols for Epidemics

1st Arvin Hekmati
CS Department
University of Southern California
Los Angeles, United States
hekmati@usc.edu

2nd Gowri Ramachandran
ECE Department
University of Southern California
Los Angeles, United States
gsramach@usc.edu

3rd Bhaskar Krishnamachari
ECE Department
University of Southern California
Los Angeles, United States
bkrishna@usc.edu

***Abstract*—Public health agencies advocate the use of contact tracing procedures to deal with pandemics such as COVID-19 to prevent the infection of a vast population. Although several mobile applications have been developed previously for contact tracing, they typically require collection of privacy-intrusive information such as GPS locations, personal data, or require infrastructures such as WiFi APs. In this paper, we introduce CONTAIN, an early proposal for privacy-sensitive contact tracing. CONTAIN is a privacy-oriented bluetooth-based mobile digital contact tracing framework that does not rely on any infrastructure-based location sensing, nor the continuous logging of personally identifiable information. The goal of CONTAIN is to allow users to determine with complete privacy if and when they have been within a short distance of someone that is infected. We identify and prove the privacy guarantees provided by CONTAIN. We also present a simulation study utilizing an empirical trace dataset which shows that users can maximize their possibility of identifying if they were near an infected user by turning on the app in more crowded settings.**

***Index Terms*—Privacy, Contact Tracing, COVID-19, Bluetooth, Mobile application**

I. Introduction

The ongoing COVID-19 epidemic has been one of the biggest health crisis the world has seen in a very long time, with more than 90 million infected individuals and more than 2 million deaths to date. The World Health Organization has long issued guidelines to limit such outbreaks by tracking people who may have come in contact with an infected individual [1]. Such measures allow government and health care officials to drastically reduce the spread of the virus to a large community. The infected individual is required to share all his or her travel details with the health care authorities to reliably track and quarantine people who may contract the virus due to their physical proximity with the infected individual. With the pervasive availability of smartphones, newer mobile-based digital contact tracing applications have emerged.

The contact tracing process typically involves gathering privacy sensitive information from the infected individual. Individuals concerned about privacy may not be willing to share all the information, which may hamper the contact tracing process while exposing populations to continued spread. A privacy-sensitive contact tracking approach could potentially encourage more people to participate in the contact tracing process confidently. This is the focus of this work.

We present CONTAIN (an acronym coined from "CONtact TrAcINg"), which is a mobile application to enable privacy-sensitive contact tracing for epidemics. We show how to leverage the users' mobile devices sending anonymous encrypted or random messages to each other via Bluetooth to allow users to determine with a 100% privacy if and when they have been in range of an infected user recently. There is no collection of privacy sensitive GPS, no reliance on external infrastructure for location tracking, no continuous logging of personally identifiable information from users. Specifically, we introduce two privacy-friendly contact tracing protocols for CONTAIN; The first protocol uses symmetric key encryption, while the second protocol requires the secure generation of sufficiently large random numbers. In a way, they are duals of each other: the first protocol nodes upload the messages they have heard, while in the second protocol, the nodes upload the messages they have sent. Both protocols provide privacy for users that are not infected as such users never need to reveal their identity or contacts. Both protocols offer users with opt-in measures to get verified and declare if they are infected, while still maintaining some measure of privacy for the general public as they do not need to reveal their true identities or those of their contacts or the locations they have visited to everyone. And both protocols allow users to privately detect if they have had contacts with others that are infected.

Our approach, particularly the randomized beacon protocol, has a lot in common with other independently proposed privacy-sensitive contact tracing protocols such as DP3T [2] and PACT [3], as well as the COVID-19 exposure notification system developed and deployed worldwide jointly by Apple and Google. However, to our knowledge, the first public description of the protocols described in this paper predates all these other works – it was first published on the web in the form of a medium article on March 4, 2020, see https://link.medium.com/qQ4k2SpyW7.

The rest of the paper is structured as follows: Section II presents the related work. The system description, design goals, and the contact tracing protocols of the CONTAIN

978-3-903176-32-4

application are presented in Section III. We present the privacy analysis of CONTAIN in Section IV. Section V describes our evaluation study and presents the key findings. Section VI concludes the paper.

II. Related Work

A number of mobile phone based contact tracing applications have been presented in the literature [4]–[7]. Qathrady *et al.* [4] and Reddy *et al.* [5] are some early works that addressed contact tracing using mobile devices, but without a focus on privacy. EPIC [6] and ENACT [7] were developed to trace contacts using a mobile application in a privacy-preserving manner, but both rely on Wi-Fi access points, wherein the mobile phones are expected to report the access point identifier and the timestamp to a server, which is then matched against the infected user's information to detect contact proximity.

Recently, a number of contact tracing solutions [2], [3], [8], [9] have been developed and deployed in various countries including Singapore, Israel, and Australia. The Singapore government deployed TraceTogether [8], which uses Bluetooth beacons to identify contacts. When the user gets infected, she/he is required to upload their local contact records to a central server. Similarly, COVIDSafe [9], which is deployed by the Australian government, also follows the same approach. Both TraceTogether and COVIDSafe uses the mobile numbers of the infected and at-risk users, which could be classified as personally identifiable information.

Our proposed protocols in this paper have a lot in common with other recently-proposed, independently-developed privacy-sensitive digital contact tracing protocols. DP3T [2], an abbreviation of "Decentralized Privacy-Preserving Proximity Tracing", is also a Bluetooth-based contract tracing protocol, which broadcasts an ephemeral identifier to mobile devices in the vicinity. When the user gets infected, D3PT allows the user to upload the identifiers broadcasted from the user's phone to a server, similar to our random beacon protocol. PACT [3] is another mobile-based contact tracing protocol with similar goals, which uses a hashed ID based on a random seed generated by the user. In early April 2020, Apple and Google first announced their joint effort to develop a privacy-sensitive digital contact tracing / exposure-notification system whose underlying protocol is based on the same approach as the random beacon protocol we describe here. To our knowledge, the earliest description of our protocols was published before these works, as a Medium article. Further, the encryption-based protocol described in this paper is different from these other approaches.

III. CONTAIN: Privacy-oriented Contact Tracing Protocols

A. Design Goals

The design goals of the CONTAIN framework are described in this section.

- The mobile application should not rely on external infrastructure such as WiFi access points with known locations or the use of GPS for determining contact information as it may violate the location privacy of the user.
- The application should not disclose any personally identifiable information to other mobile devices.
- A user should be able to check if they have been near an infected user on their own, in a completely private manner, without being forced to reveal to anyone else their infection status.
- The anonymized information needed for other users to determine if they have been in contact with a user that is infected should be made available to the public through a trusted server only if the user can prove to the operator of the server (e.g. through a medical certificate) that he or she is infected.

The components of the CONTAIN framework are discussed below.

- **Mobile Device** The core processes of the CONTAIN application run on user's smartphones (mobile devices). We describe below two different protocols that can be used for the CONTAIN application, one based on encrypted beacons (see Section III-B) and one based on random beacons (see Section III-C). The CONTAIN application does not rely on or require GPS (in a given implementation of it, it may be possible to additionally allow users to reveal GPS information to the application on an opt-in basis, but this is not an essential part or requirement of our design).
- **Bluetooth technology** Bluetooth is one of the widely used communication technology in contemporary smartphones. Today, this technology is used for short-range communication with hands-free earphones and external Bluetooth-based devices such as speakers and other audio systems. The communication range of Bluetooth is on the order of 10 meters or less [10].
- **Verification server** The smartphone and the Bluetooth technology can be used to gather contact traces in a private manner, which is discussed in Section III-B and III-C. A third-party server is required to verify the infection status of a user (than claims to be infected) and then to make anonymous information from that infected user available for other users to use, in a private manner, to verify if they have been near the infected user.

with computation resources and ability to access the verification server over the Internet (through any means such as Cellular or WiFi). Note that the contact tracing logic on the phone transmits and receives an anonymous encrypted or random Bluetooth beacon from other mobile devices running the CONTAIN application.

B. Protocol 1: Encrypted Beacons

Figure 1 shows the illustration of the first contact tracing protocol.

1) Each user periodically beacons, using Bluetooth, a message, $\mathcal{M}$, consisting of a unique name or ID, a timestamp, and a random number (salt) that changes over

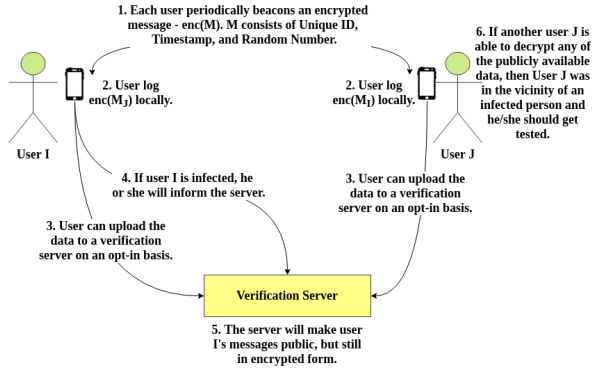

Fig. 1. Illustration of how the Privacy-Sensitive Contact Tracing App would work with our first protocol.

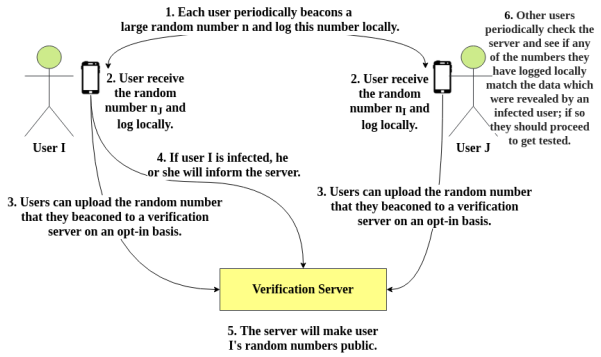

Fig. 2. Illustration of how the Privacy-Sensitive Contact Tracing App would work with our second protocol.

time. Each message is encrypted using a symmetric key $enc(\mathcal{M})$. Note that the users do not share this key with others.

2) Other users that hear the encrypted message beacon log it locally.
3) Either periodically or in a batch, on an opt-in basis, each user can upload all the encrypted beacon messages they have heard to a common verification server.
4) If a user i becomes infected, they inform the server, on an opt-in basis, with evidence that they are infected, such as a medical report or "infection certificate". The above step of uploading encrypted beacon messages could also be taken in conjunction with this step.
5) The verification server proceeds to make all the messages uploaded by user i publicly available, but still in the encrypted form.
6) Each other user j can privately check these now publicly available encrypted messages to see if they can decrypt any of them. Note that this verification process could be automated. If any user manages to decrypt the message, then he or she is at risk as he or she has been near an infected person. In this case, the user should then proceed to get tested themselves.

C. Protocol 2: Random Beacons

Figure 2 shows the illustration of the second contact tracing protocol.

1) Each user beacons a sufficiently large random number and logs this random number locally. Here, a large number is chosen to minimize the chance of collisions with beacons generated by other users.
2) Others that hear the beacon log this number locally.
3) Either periodically or in a batch, on an opt-in basis, each user can upload all the random beacon messages they have transmitted to a common verification server.
4) If a user i becomes infected, on an opt-in basis, they inform the verification server with evidence that they are infected, such as a medical report or "infection certificate". The above step of uploading the random beacon messages could also be done in conjunction with this step.
5) The verification server then proceeds to make all the random numbers uploaded by this user publicly available.
6) Other users periodically check the server and see if any of the numbers they have logged locally match those that were revealed by an infected user; if so they should proceed to get tested.

The fourth step in both protocols requires a trusted third party that a) operates the "publicizing" server, and b) can verify through a medical certificate that the user whose uploaded anonymous data is being made public has truly tested positive for being infected. This party must be trusted and perform the verification correctly to minimize false alarms. It may be necessary for this role to be fulfilled by an organization (whether public or private) with relevant experience and reputation. For example, it could be a healthcare organization or government entity, or medical insurance provider.

Both protocols require the ability to beacon information over short range Bluetooth links. Today's mobile phones, for security, privacy and energy efficiency reasons, generally require applications that do such beaconing to run in the foreground; regrettably, this could make the protocols less effective than they could be. But perhaps mobile device manufacturers could be prevailed upon to provide greater access to such an app in the public interest. Alternatively, at least at relatively more crowded venues such as airports, campuses, shopping areas or public transportation, users can be encouraged to download and turn on these apps.

D. Anonymizing an Infected User

Note that the information revealed by the verification server does not explicitly contain anything that would reveal to others who the infected user is; it only allows other users to determine if they have been near an infected user and when. In some cases, if a user knows that during a certain time they were only in contact with (within bluetooth range) of a specific person they know, they may be able to infer the identity of the infected person from the time information provided. One way the infected user could potentially avoid even this minimal

chance of disclosure would be to only release information from times when they know they were in the presence of many other devices (which could be determined based on the timestamps of logged messages), and of course they could also opt-out of notifying other parts.

While it may be assumed the infected user must make themselves known to the owner/operator of the verification server since they must provide the medical certificate that they are infected, there is an additional step that could be taken to anonymize them further. This would involve asking the infected user to use a public-key pair that does not tie them to a real-world identity, and have the medical authority digitally sign an infection certificate (using their own key pair) that only includes the infected user's public key. The infected user can then provide this digital certificate to the verification server. From this point on the verification server can verify using the public key of the medical authority that the anonymous user contacting it (whom they know only through its public key) is indeed infected, and the verification server can thus ensure that only the (anonymous) data uploaded by a user that is truly infected is being made publicly available.

E. Anonymizing Encounter Time

We would like to clarify one additional aspect that is *not* guaranteed to be kept private by either protocol. If an infected user i has opted-in to notify the verification server and share the data it logged, then any user j that has actually been near the user i will know the time of encounter, which will a) de-anonymize the location of the encounter (since the user may remember where they were at that encounter time; or *a fortiori* in protocol 1 they could also inject their own GPS location explicitly into the message that is encrypted to keep track of where they were during each encounter), and b) de-anonymize the identify of the infected user i to j (if user j knows and remembers or has some other way of identifying who they were near at the time of encounter).

Fortunately, there is a solution to this problem. If it is desired to further strengthen the system to provide for an additional layer of privacy to avoid the above shortcoming, one possible solution would be to use protocol 2 (random beacons) and eliminate the public notification step so that the verification server does *not* publish all the random numbers sent by the infected user. Instead, require the user interested in checking if they were near an infected user to upload all the random numbers they have logged along with an anonymizing "ID" to the server (using TOR or something similar to ensure an anonymous upload). Then the server will verify if there is any overlap between the random numbers from the infected user and random numbers uploaded the user that is checking. If so, it publishes only a notification that the user corresponding to that anonymous ID has been (or not been) in contact with an infected user.

IV. Privacy Analysis

In this section we formally identify and prove the privacy guarantees we can provide. The properties and their proofs assume that the trusted verification server behaves correctly in all cases.

We first list the key privacy properties that both our protocols guarantee below:

- **P1.** Beacons emitted by a user do not reveal any personally identifying information or location information about that user to other users.
- **P2.** Users that are not infected are not required to upload any information to the verification server.
- **P3.** Users will not be notified by the verification server about or receive any data from the verification server that can help them verify potential contact with users that are not infected
- **P4.** Users that are infected must opt-in in order for other users to determine if they have been near that infected user.
- **P5.** Users can check if they have been near an (opted-in) infected user without revealing any personally identifiable information about themselves.

Further, there are two additional privacy properties that can be shown to hold with the appropriate additional modifications as discussed in section III-D (for both protocols) and section III-E (for protocol 2 only):

- **P6.** Users that are infected can keep themselves anonymous even from the operator of the verification server.
- **P7.** A user that finds out it has been contact with an infected user will not be able to determine when or where exactly the contact happened.

Proof of P1: We assume that the underlying Bluetooth protocol itself is making user of ephemeral, time-varying ID's and not sending static, identifiable MAC addresses (such schemes are already implemented by privacy-sensitive mobile device operating systems such as Apple's iOS [11]). In protocol 1, the beacons are encrypted using a symmetric key that is not revealed to any other user. The use of a salt that can be changed each time further makes it so that there is no way to connect a particular individual or device with the logged data since it will be different each time the salt is changed. In protocol 2, the beacons consist of random numbers that can also change each time, so again, there is no identifying information being transmitted. Neither protocol requires collection of any location information. In protocol 1, the user could chose to encrypt their own GPS data if they so wish, but this information could only be checked by them on their own since it is encrypted with a key that only they possess.

Proof of P2: This property is a direct consequence of the two protocols. At no step is a device required to upload any information if they are not infected. Although step 3 in each protocol does allow each user to upload their random logged info periodically, they are not required to do so before they are infected, and even then it is on an opt-in basis.

Proof of P3: Since users that are not infected do not upload any information to the verification server in either protocol,

there is no way for other users to be notified about or learn anything about them.

Proof of P4: For other users to determine if they have been near an infected user, the verification server needs to make available logs from that infected user. However, in both protocols, the logs are provided by infected users only on an opt-in basis.

Proof of P5: For both protocols, the verification server can make the anonymous information they get from the infected users available on a public website. Users can use an anonymous web browser such as TOR to download that information and check if they have been near any of the infected user in a completely private manner, without revealing any of their identifying information to the operator of the verification server or any other party.

Proof of P6: To allow infected users to keep themselves anonymous from the verification, the certificate from the medical authority should include only the public key of the infected user, as described in the additional step of section III-D. If that additional measure is taken, then the infected users gain an additional level of anonymity and privacy.

Proof of P7: This is addressed by the solution presented in Section III-E. By restricting to protocol 2 and asking users that want to check if they are infected to anonymously provide their logged data to the verification server, it can be ensured that the users that are checking will be notified that there was an encounter but not when or where the encounter happened.

V. Evaluation

Most mobile operating systems today require apps that use Bluetooth communication to run in the foreground. Thus it requires active user intervention to be useful. In this section we evaluate through trace-based simulations how well CONTAIN would work as a function of how long the app is turned on by the user each day, and when it is turned on[1]. We use a dataset named Asturies provided by CRAWDAD.org for our simulation [12]. The dataset contains the encounter of mobile devices with each other at different timestamps. We split the day time into 24 slots and mapped one hour of the dataset to each time slot. Users are considered to turn on the CONTAIN mobile app to participate in the experiment according to three different scenarios, which are described in Section V-A.

During the experiments, we varied both the number of initial infected users that are chosen randomly and also the probability that the disease gets transferred between people to see their effects on the results. Moreover, we considered several time slots that users turn on their Bluetooth-based CONTAIN app to evaluate the effectiveness of our protocols in different scenarios. Each experiment is repeated 1000 times to capture the randomness of the simulations.

A. Methodology

We conducted two sets of experiments. First we designed an experiment to study how the infection rate grows among people according to the initial number of infected users and the contagiousness probability.

Second, we designed an experiment with three different scenarios in terms of when users schedule the time slots that they turn on their Bluetooth-based CONTAIN app.

- **Random:** In this scenario, users turn on their Bluetooth randomly in 12 contiguous hours that correspond to the most active time of the day.
- **Decentralized:** In this scenario, each user turns on his or her Bluetooth and runs the CONTAIN app when they are in a crowded place such as train stations, offices, shopping malls, etc. in a decentralized manner. For this scenario, we sort the time slots for each user according to the number of devices that the user connected to in that time slot. The second criterion for sorting is the crowdedness of the area that the user is. We measure the crowd level of an area by counting the number of records that we have in each time slot. Then, during an experiment where the user turns on for k slots, it does so for the first k slots in the sorted order.
- **Centralized:** In this scenario, we assume that there is some centralized coordination mechanism that ensures that all users turn on their Bluetooth and run the CONTAIN app at specific hours of the day. In this scenario, we sort the time slots according to first the average number of active mobile devices, and second the number of records that we have in each hour in the dataset. Thus the app is activated by everyone during the hours with the most active mobile devices.

B. Results

Figure 3 (left) shows the results from our first set of experiments, showing the number of users who get infected by infectious diseases such as CoVID-19 and SARS-CoV2 with 90% confidence interval versus contagiousness probability. These are also the number of initial users who are assumed to be infected at the beginning of the simulation. As you can see, as we have more number of initially infected users and also higher contagiousness probability, more people get infected.

Figure 3 (right) represents the number of users who have been in contact with or in the proximity of infected persons. In this experiment, we assumed the contagiousness probability is 2%, and the initial number of infected users is 2. From Figure 3 (right), it is clear that as more people cooperate and turn on their Bluetooth for more number of time slots, more number of potentially infected users could be identified reliably.

As we can see in Figure 3 (left), when the number of infected people grows so fast, it is essential to identify and notify potentially infected people to get tested. Besides, this helps the individuals who may have come in contact with the infected individual to take corrective actions, including self-isolation, to prevent him or her from spreading the infection to others.

In this experiment, we have, on average, 100 devices active per week and around 60000 records for those devices per

[1]Note that if the protocols are enabled directly by Mobile OS providers such as the widely deployed Apple-Google exposure notification system, then users need not intervene and the application can run entirely in the background

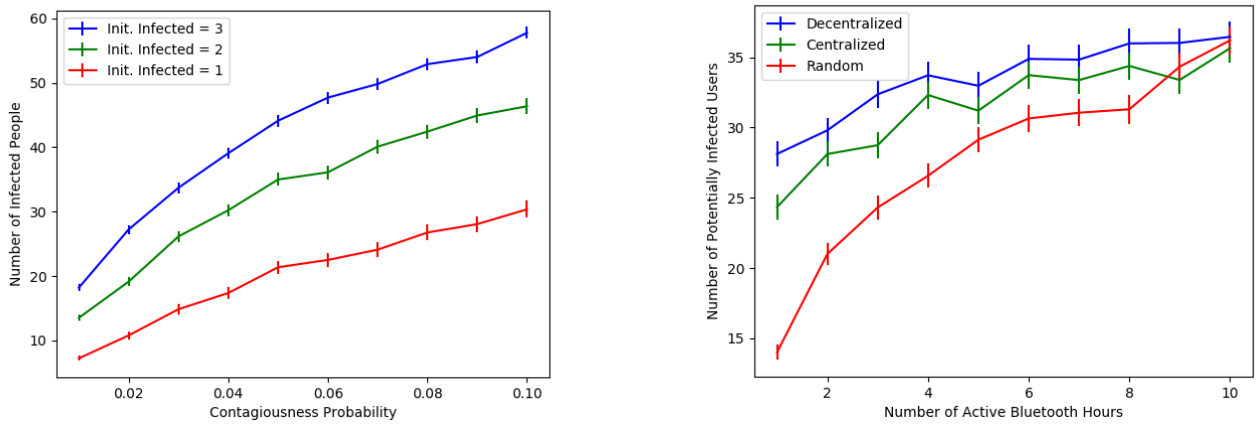

Fig. 3. Left: Number of Infected Users vs Contagiousness Probability and Initially Infected Users; Right: Number of Test Required Users vs Number of Active Bluetooth Hours

week, which indicates a dense environment with lots of interactions among people. That is why such a small contagiousness probability and also initially infected people results in a considerable number of infected users. Consequently, as we can see in Figure 3 (right), our application informs many more people in case more number of them actively use the Bluetooth-based CONTAIN application. In this way, people can be alerted to get tested, take care of themselves and not spread the virus among other people by restricting their activities and avoiding crowded areas.

It is also interesting and important to note that when people act in a decentralized manner and turn on their Bluetooth-based app only when they go to a crowded environment, we will also have better performance as compared to the centralized and random scenarios. It is also interesting to see that decentralized scenario results in more number of people being detected as potentially infected and required to do the test as compared to the centralized scenario. This is because, if each person turns his or her Bluetooth in the places where lots of other people are available (this is evident from the dataset that we used in our study), much more number of encounters will be recorded as compared to the centralized rule, which does not fit everyone's activity.

VI. Conclusion

In this paper, we have presented CONTAIN, a privacy-friendly and Bluetooth-based mobile application for contact tracing. We have presented two contact tracing protocols, both of which do not reveal any personally identifiable information. Through a simulation study, we have evaluated how well CONTAIN can identify infected users under different activation regimes. However, this study assumes that users must actively turn the application on in order for the protocol to work – a better solution is for mobile OS developers to provide support for the protocols in an inter-operable manner so that users can opt-in and the app will run in the background. This is precisely the approach that has been adopted by Apple and Google in their exposure notification protocol, which is now widely deployed.

References

[1] W. H. Organization *et al.*, "Contact tracing during an outbreak of ebola virus disease," 2014.

[2] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli *et al.*, "Decentralized privacy-preserving proximity tracing," *arXiv preprint arXiv:2005.12273*, 2020.

[3] J. Chan, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, S. Singanamalla, J. Sunshine *et al.*, "Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing," *arXiv preprint arXiv:2004.03544*, 2020.

[4] M. A. Qathrady, A. Helmy, and K. Almuzaini, "Infection tracing in smart hospitals," in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2016, pp. 1–8.

[5] E. Reddy, S. Kumar, N. Rollings, and R. Chandra, "Mobile application for dengue fever monitoring and tracking via GPS: case study for fiji," *CoRR*, vol. abs/1503.00814, 2015.

[6] T. Altuwaiyan, M. Hadian, and X. Liang, "Epic: Efficient privacy-preserving contact tracing for infection detection," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[7] A. Prasad and D. Kotz, "Enact: Encounter-based architecture for contact tracing," in *Proceedings of the 4th International on Workshop on Physical Analytics*, 2017, pp. 37–42.

[8] J. Bay, J. Kek, A. Tan, C. S. Hau, L. Yongquan, J. Tan, and T. A. Quy, "Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders," *Government Technology Agency-Singapore, Tech. Rep*, 2020.

[9] C. App, "Australian government department of health: Covidsafe app," *Last accessed: 29th April*, 2020.

[10] M. Collotta, G. Pau, T. Talty, and O. K. Tonguz, "Bluetooth 5: A concrete step forward toward the iot," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 125–131, July 2018.

[11] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown, "A study of mac address randomization in mobile devices and when it fails," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 365 – 383, 2017.

[12] S. Cabrero, R. García, X. G. García, and D. Melendi, "CRAWDAD dataset oviedo/asturies-er (v. 2016-08-08)," Downloaded from https://crawdad.org/oviedo/asturies-er/20160808, Aug. 2016.