Mind the GAP: Security & Privacy Risks of Contact Tracing Apps

Lars Baumgärtner*[1], Alexandra Dmitrienko[3], Bernd Freisleben[2], Alexander Gruler[2], Jonas Höchst[1,2], Joshua Kühlberg[1], Mira Mezini[1], Markus Miettinen[1], Anel Muhamedagic[1], Thien Duc Nguyen[1], Alvar Penning[2], Dermot Frederik Pustelnik[1], Filipp Roos[3], Ahmad-Reza Sadeghi[1], Michael Schwarz[2], and Christian Uhl†[2]

[1]TU Darmstadt, Germany
[2]Philipps-Universität Marburg, Germany
[3]JMU Würzburg, Germany

June 9, 2020

*List of authors in alphabetical order

†Corresponding authors' email: {markus.miettinen, ahmad.sadeghi}@trust.informatik.tu-darmstadt.de; {baumgaertner, mezini}@cs.tu-darmstadt.de; {filipp.roos, alexandra.dmitrienko}@uni-wuerzburg.de; {hoechst, freisleb}@informatik.uni-marburg.de

Abstract

Contact tracing apps running on mobile devices promise to reduce the manual effort required for identifying infection chains and to increase the tracing accuracy in the presence of COVID-19. Since the beginning of the pandemic, several contract tracing apps have been proposed or deployed in practice by academia or academic-industrial consortia. While some of them rely on centralized approaches and bear high privacy risks, others are based on decentralized approaches aimed at addressing user privacy aspects. Google and Apple announced their joint effort of providing an API for exposure notification in order to implement decentralized contract tracing apps using Bluetooth Low Energy, the so-called "Google/Apple Proposal", which we abbreviate by "GAP". The contact tracing feature seems to become an opt-in feature in mobile devices running iOS or Android. Some countries have already decided or are planning to base their contact tracing apps on GAP[1].

Several researchers have pointed out potential privacy and security risks related to most of the contact tracing approaches proposed until now, including those that claim privacy protection and are based on GAP. However, the question remains as how realistic these risks are. This report makes a first attempt towards providing empirical evidence in real-world scenarios for two such risks discussed in the literature: one concerning privacy, and the other one concerning security. In particular, we focus on a practical analysis of GAP, given that it is the foundation of several tracing apps, including apps such as the Swiss *SwissCOVID*, the Italian *Immuni*, and the German *Corona-Warn-App*. We demonstrate that in real-world scenarios the current GAP design is vulnerable to (i) profiling and possibly de-anonymizing infected persons, and (ii) relay-based wormhole attacks that principally can generate fake contacts with the potential of significantly affecting the accuracy of an app-based contact tracing system. For both types of attack, we have built tools that can be easily used on mobile phones or Raspberry Pis (e.g., Bluetooth sniffers). We hope that our findings provide valuable input in the process of testing and certifying contact tracing apps, e.g., as planned for the German *Corona-Warn-App*, ultimately guiding improvements for secure and privacy-preserving design and implementation of digital contact tracing systems.

1 Introduction

Caused by coronavirus SARS-CoV-2, the COVID-19 disease spreads particularly through direct contact between people. Health authorities face the challenge of identifying and isolating infection chains to prevent the pandemic from further spreading. This task usually involves manual effort and relies on contact information that is voluntarily provided by infected people. Hence, infection chains have to be reconstructed by the health authorities with an enormous amount of work for each individual case, but nevertheless they are not always accurate or complete.

Using advanced digital contact tracing apps on mobile devices can help to reduce the manual effort and significantly increase the tracing accuracy. This has already been demonstrated in Asia (e.g., Singapore, China, Korea). Even if - for understandable reasons - there is still a lack of empirical evidence about the actual role that contract tracing apps played in fighting the pandemic in these countries compared to other measures, such as massive testing and manual tracing, the call for contact tracing apps became urgent in the rest of the world that was hit by the pandemic later.

The recent "arms race" on providing a contact tracing app has created various mobile contact tracing approaches and corresponding apps, published in Github repositories, or even deployed in countries such as China, South Korea, Singapore, Taiwan, Austria, Australia, France, Italy, Switzerland, and UK. Furthermore, several active development efforts are currently running in Europe (e.g., Germany) and in the US (e.g., MIT, UC San Diego). For an overview and a comparison of contact tracing apps, we refer to Miettinen et al. [18]. Apart from proposals made by academia, IT enterprises such as Microsoft, Apple, and Google are active in working on contact tracing apps. In an unprecedented joint effort, Google and Apple announced that they will provide an Application Programming Interface (API) for exposure notification at the mobile operating system level in order to implement contract tracing apps using Bluetooth Low Energy (Bluetooth LE). We call this the *Google/Apple Proposal* [2], abbreviated by GAP in the rest of the paper.

[1] `https://gizmodo.com/23-countries-are-already-onboarding-the-apple-google-co-1843587139`

The contact tracing technologies used in a number of Asian countries collect highly sensitive data from individuals. However, data protection and privacy regulations, especially with respect to medical data, may significantly differ from country to country. For example, in several European countries or in the US, there are more restrictive regulations. Hence, in Europe and the US, researchers and governmental agencies are investing significant efforts to develop digital approaches to contact tracing that can provide an appropriate level of security and privacy and to make the development process somehow transparent. For instance, the development of both the Swiss and the German apps are following an open source approach, enabling the community to be involved in early testing of the app code, at least in theory.

Yet, scepticism remains regarding several aspects of contact tracing apps and the corresponding infrastructure. Some people go so far as seeing contact tracing apps not as a solution to the problem, but as a part of the problem itself[2]. For a highly interesting discussion on these aspects, we refer to a recent opinion statement by Ross Anderson [1]. From a more technical perspective, several researchers have pointed out the possibility of profiling infected persons in the GAP approach [5, 7, 12], as well as the possibility to perform relay attacks [5, 9, 23, 28, 29]. The goal of this work is to perform a reality check towards providing empirical real-world evidence for the above two privacy and security risks discussed in the literature.

We focus on approaches that use Bluetooth LE for sensing proximity between users. In particular, we present a summary of our practical analysis of the GAP. We selected the GAP approach for the following reasons. First, GAP will be broadly adopted, since several European contact tracing apps, such as the Swiss *SwissCOVID*, the Italian *Immuni*, and the German *Corona-Warn-App*, are based on the GAP API. Second, the GAP API is already opt-in for iOS and Android devices, hence, it will potentially stay with us for a long time. Moreover, the GAP API implementation is currently closed-source, thus, we cannot be confident that important and sensitive health-related data is not going to be used in other ways.

We demonstrate that in real-world scenarios the current GAP design is vulnerable to (i) profiling and possibly de-anonymizing infected persons, and (ii) relay-based wormhole attacks that principally can generate fake contacts with the potential of significantly affecting the accuracy of an app-based contact tracing system. For both types of attack, we have built tools that can be easily used on mobile phones or Raspberry Pis (e.g., Bluetooth sniffers). We hope that our findings provide valuable input in the process of testing and certifying contact tracing apps, e.g., as planned for the German *Corona-Warn-App*, ultimately guiding improvements for secure and privacy-preserving design and implementation of digital contact tracing systems.

2 Privacy & Security Risks of Contact Tracing Apps

2.1 Privacy Risks

While in some countries privacy aspects did not receive particularly high attention, there has been a lively debate around privacy compliance in the EU and US. The crucial risks associated with contact tracing apps are related to the potential misuse of the information they collect. Compared to many other types of applications, an app for contact tracing inevitably collects an exceptionally large amount of sensitive information about the contacts and relationships between individual users in great detail. In addition, for a mobile app to be effective for its intended purpose, as many citizens as possible should voluntarily install the app and actively use it. For this reason, the system infrastructure related to contact tracing apps will collect vast amounts of information about the mutual contacts of people – considerably more than any other approach currently used by health authorities! Therefore, the mobile app and the back-end server infrastructure must be designed and implemented in such a way that effective contact tracing can be achieved, while minimizing the exposure of privacy-sensitive user data.

It is noteworthy to mention that people often argue that their mobile phones and service providers already know a lot about them. So, what difference would a more privacy-preserving contact tracing app make after all, since the 'game is lost' anyway? We strongly believe that if we follow this line of reasoning about our privacy, then we also have to accept the potential misuse

[2] www.brookings.edu/techstream/inaccurate-and-insecure-why-contact-tracing-apps-could-be-a-disaster

of information when individuals with diseases apply for jobs, ask for loans, and look for insurance contracts, etc.

On the contrary, people should have the right to control who, when, and where someone has access to their data. The less private information users disclose to these providers, the better their privacy can be protected. Furthermore, in contact tracing scenarios we are concerned with possibly infected people who may not want to disclose this information to anyone they do not trust. We believe that many people would agree that medical data is private information that should not be disclosed to others without the consent of the concerned individual.

2.1.1 Contact Tracing via GPS

Some approaches are based on tracking the GPS location of participating users [21, 26]. However, the use of GPS for this purpose faces several challenges, since GPS is relatively inaccurate especially in indoor areas that are particularly important for capturing contacts accurately due to the higher contagion risk in enclosed spaces. User privacy is addressed in these approaches by allowing users to redact locations that they deem sensitive. However, this approach has its problems. For example, many potential contacts are lost when places like homes and workplaces are redacted from released location traces, thus diminishing the effectiveness of the system. On the other hand, even aggressive redaction of specific locations may not be sufficient for ensuring user privacy, since users may still be identifiable given additional information that, e.g., social media companies or players like Google have on their users.

2.1.2 Contact Tracing via Bluetooth LE

By recording proximity identifiers observed by Bluetooth LE sensors in a number of strategically placed observation points in a given area (e.g., in a particular city), it is possible to track the movements of individual users between observation points and thus collect detailed movement profiles of individual users.

Although contact tracing systems do not explicitly collect or record the true identities of individual users, movement profiles based on pseudonymous tracing data make it possible to identify a large fraction of users with high probability. This is mainly because movement profiles are quite distinctive. For example, using only the locations of home (main location during the night) and the workplace (main location during the day) makes it possible to identify a large number of people unambiguously. Furthermore, several other possibilities for de-anonymization have been proposed. For example, by placing a Bluetooth LE sensor close to a camera with facial recognition functionality, it is in principle possible to directly associate the proximity identifier beaconed over Bluetooth LE (and thus the used pseudonym) with an identifiable person to entirely de-anonymize the person in question.

2.1.3 Centralized vs. Decentralized Contact Tracing

Centralized contact tracing models, such as the TraceTogether app used in Singapore [11] or the CovidSafe app used in Australia [3], as well as the PEPP-PT model originally planned to be deployed in Germany [6], the StopCovid app in France [20], and the NHS COVID-19 app in the United Kingdom [19] do not provide adequate privacy protection for their users.

These approaches are based on the principle that a centralized back-end server system assigns a unique identifier (a pseudonym) to each user's mobile app, from which frequently-changing pseudonymous proximity identifiers are derived and broadcast over Bluetooth LE to other nearby devices. The weakness of this approach is that the back-end server system can associate all proximity identifiers of all users with the unique pseudonym of each user. This in turn allows operators of the back-end server system to perform comprehensive monitoring of all users of the system.

Due to these severe weaknesses, contact tracing apps should be based on decentralized identification of contacts. In the decentralized model, the back-end server system does not have information about proximity identifiers of users and therefore cannot associate them with individual users. Indeed, due to the fundamental problems associated with the centralized contact tracing model and the huge resistance of the scientific community[3], the Federal Government of Germany decided to

[3] `https://drive.google.com/file/d/1OQg2dxPu-x-RZzETlpV3lFa259Nrpk1J/view`

abandon the centralized approach of the PEPP-PT consortium that it initially highly supported, and decided to switch to a decentralized contact tracing approach.

2.2 Security Risks

Privacy is only one part of the story. The other part is concerned with security risks that contact tracing apps may introduce, which may affect individuals, companies, institutions, and the contact tracing systems as a whole. For example, relay attacks in the context of contact tracing apps have been discussed in a recent EU e-Health Network whitepaper [9], as well as in the context of PACT [5] and DP-3T [23, 28, 29]. Gvili [12] proposes countermeasures, such as time- and (geographical) cell-based verification of received messages or the need for bidirectional communication. Beskorovajnov et al. [4] argue that every contact tracing protocol that does not incorporate a handshake mechanism or validation of time or location is vulnerable to relay attacks.

Unfortunately, countermeasures to avoid relay attacks that require location information do not only bear privacy risks, but also lack practical usefulness, because location information, e.g., GPS in an urban environment, is often inaccurate [17] and mostly useless in indoor situations where Bluetooth LE contact tracing is supposed to show its value: examples are department stores, restaurants, trade fairs, conference venues, concert halls, indoor sports events, airports, and airplanes. Furthermore, countermeasures based on handshake mechanisms or validations of time would go against the idea of minimizing information to be exchanged for contact tracing, and thus would probably create additional security and privacy risks.

3 Mind the Privacy GAP

Despite being designed with the principle of minimal data in mind, the GAP protocol still has issues in terms of privacy properties. The main problem is that it requires infected individuals to publish all Bluetooth LE proximity identifiers they have used during the days they may have been infectious to all devices participating in the system. Thus, this information essentially becomes public, and it is possible for all app users and other entities involved in the system to potentially track the movements of (and possibly de-anonymize [16, 25]) infected persons during these days. In the rest of this section, we present our real-world attacks on the privacy of the GAP scheme based on its specification [2].[4]

3.1 GAP Overview

The GAP contact tracing approach [2] is based on frequently-changing random pseudonyms, so-called *Rolling Proximity Identifiers (RPI)*. An overview of the approach is shown in Fig. 1. Each app generates these RPIs from a *Daily Tracing Key (DTK)* and beacons them into their surroundings using Bluetooth LE. Apps on other devices in close proximity can observe these RPIs and store them locally as a record of contact with the device beaconing the RPI. This dataset includes additional metadata like the received signal strength.

Should a user be tested positive for COVID-19, that user's app uploads DTKs of the last x days to a central server (currently $x = 14$). The server accumulates the received DTKs of infected persons and offers them to be downloaded by other users' apps. Apps in devices of participating users regularly check the server for updates and download any new DTKs. Each app then uses the downloaded DTKs to calculate the corresponding RPI pseudonyms used by the infected persons' apps in the recent past. The operating system / corresponding system service then compares these infected persons' RPIs to the RPIs stored locally on the device. If matching RPIs are found, the metadata, e.g., signal strength or duration of the encounters, related to these matching RPIs are used to calculate a risk score that is used to determine whether a warning should be displayed to the user or not.

[4] Since the GAP API is only permitted to be used by governmental health institutions [1], we could only evaluate publicly available specification documents and code.

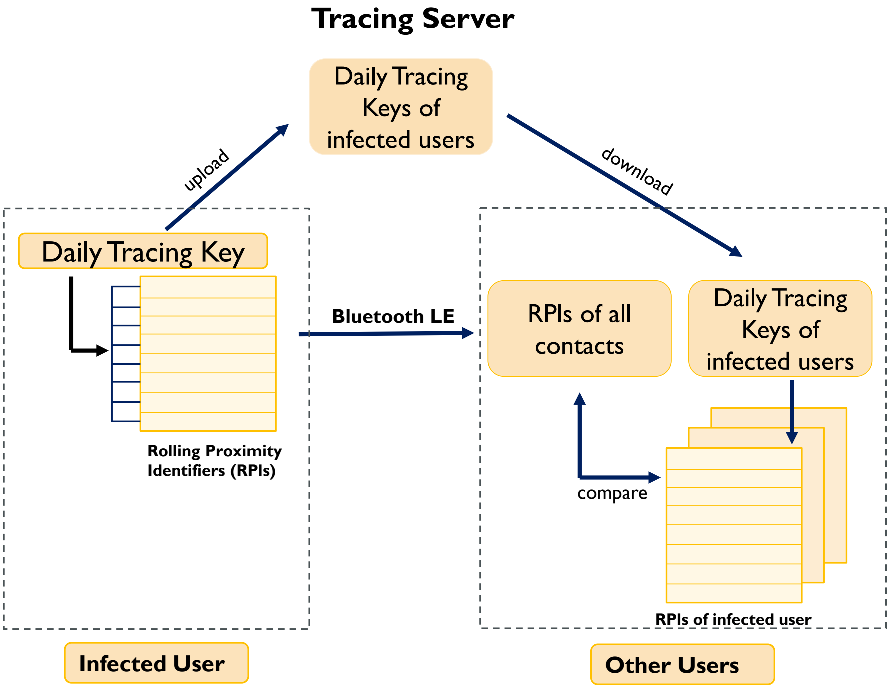


Figure 1: Overview of the GAP contact tracing approach

3.2 Privacy Attack: Profiling Infected Persons

The possibility of profiling infected persons in the GAP approach has been pointed out by several researchers [5, 7, 12]. Moreover, it is a known fact that system parameters that remain unchanged in the system for a while can also be linked together.

3.2.1 Goal and System Setup

The goal of our experiment is to show that it is practically possible to profile the movement and activities of infected users after they share their Daily Tracing Keys (DTKs) in an efficient manner. Based on DTKs, other apps participating in the system can derive the corresponding Rolling Proximity IDs (RPIs) that the infected user's app has beaconed out in the recent past. Note that since the DTKs can be downloaded by anyone participating in the system, the RPIs are essentially public information.

To conduct the attack, we deployed Bluetooth LE sniffers to capture RPIs at six targeted sensitive places downtown the city of Darmstadt, Germany, as listed in Tab. 1. The locations of these places in Darmstadt are shown in Fig. 2. We used commodity smartphones as the sniffers that can capture Bluetooth LE signals at a distance of up to 6 meters. However, with a special Bluetooth antenna it was possible to capture signals at a much higher distance. The Bluetooth LE sniffers would capture RPIs of any users moving through or spending time at the places mentioned above. In our experiment, two tracing app users simulate two particular paths.

Table 1: List of locations where Bluetooth LE sniffers were deployed

Location	Description
A	A residential area
B	City hall
C	Police station
D	Clinic and pharmacy
E	Outside a pub
F	Outside a head shop and a sports gambling bookmaker

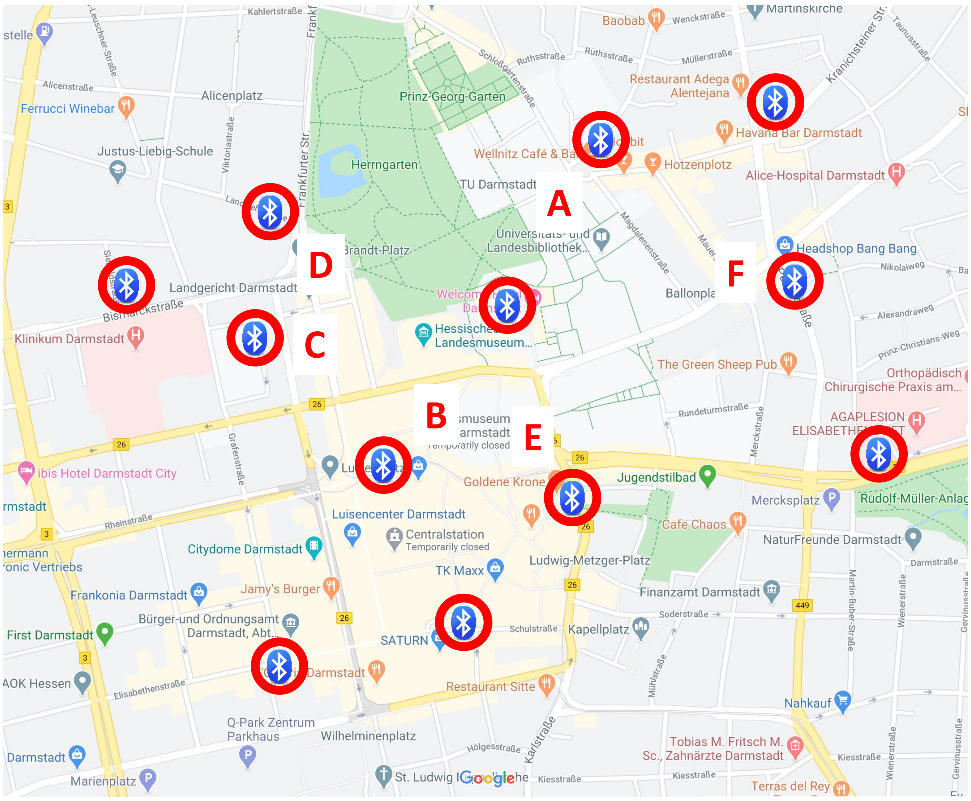


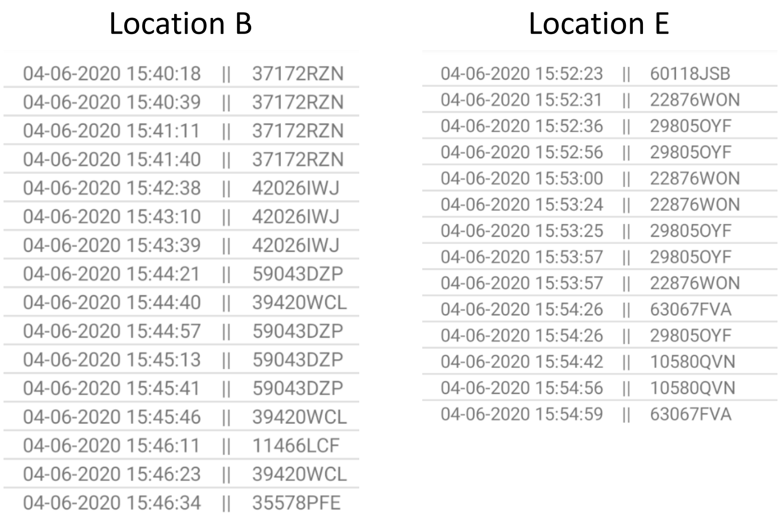
Figure 2: An example of observation points

Since the official GAP API can currently only be used by governmental health institutions (and will probably not be available to research organizations in the near future, since only officially approved apps will receive the appropriate permissions to use the API), we implemented a GAP tracing app simulator, following the RPI generation procedure laid out in the GAP cryptography API specification [2].

3.2.2 Experimental Results

A sample of our experimental results of RPI measurements captured at different observation points is shown in Fig. 3. The captured data look entirely random, and it is not obvious which RPIs could be associated with individual users. However, when we simulate the case that any one of the users is tested positive for COVID-19 and the users consequently share their DTKs that were used to derive the corresponding RPIs, a completely different picture emerges, as shown in Fig. 4. It is evident that by matching the RPIs of User 1 with the RPIs captured in different locations, e.g., location B and location E, we know exactly which locations User 1 has visited and when User 1 arrived and left each location.

Moreover, if we sort the locations that the users have visited in chronological order, we see that we can track the movements of each of the test users, as shown in Fig. 5. Let (User i, X) denote the presence of User i at location X. The sequence of observations of User 1 was as follows: (User 1, A), in a residential area, then (User 1, D) near a clinic and a pharmacy, (User 1, C) near the police station, (User 1, B) (Darmstadt city hall), (User 1, E) (near to a pub) before concluding the round at the starting point with (User 1, A), corresponding again as mentioned, to a residential area. This may potentially indicate that the user may be living in this area. A similar tracing of locations is possible for User 2 who was first observed at (User 2, B), the city hall, after which the next observation (User 2, E) happened near the pub, after which the final observation (User 2, F), was near a head shop and a sports gambling bookmaker. Using these observations about users and the associated timestamps, a significant amount of information about users can be gathered, since the attacker obtains knowledge about which places the users visit and how much time they spend



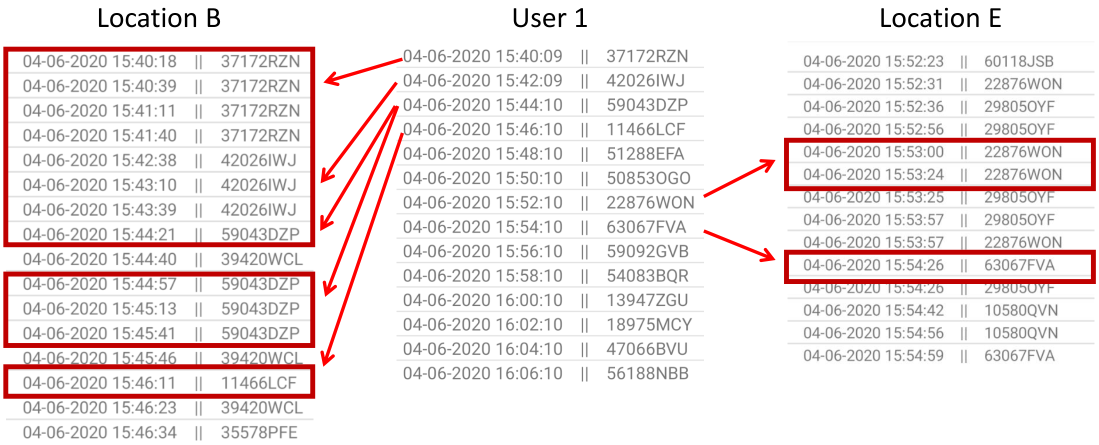
Location B

```
04-06-2020 15:40:18  ||  37172RZN
04-06-2020 15:40:39  ||  37172RZN
04-06-2020 15:41:11  ||  37172RZN
04-06-2020 15:41:40  ||  37172RZN
04-06-2020 15:42:38  ||  42026IWJ
04-06-2020 15:43:10  ||  42026IWJ
04-06-2020 15:43:39  ||  42026IWJ
04-06-2020 15:44:21  ||  59043DZP
04-06-2020 15:44:40  ||  39420WCL
04-06-2020 15:44:57  ||  59043DZP
04-06-2020 15:45:13  ||  59043DZP
04-06-2020 15:45:41  ||  59043DZP
04-06-2020 15:45:46  ||  39420WCL
04-06-2020 15:46:11  ||  11466LCF
04-06-2020 15:46:23  ||  39420WCL
04-06-2020 15:46:34  ||  35578PFE
```

Location E

```
04-06-2020 15:52:23  ||  60118JSB
04-06-2020 15:52:31  ||  22876WON
04-06-2020 15:52:36  ||  29805OYF
04-06-2020 15:52:56  ||  29805OYF
04-06-2020 15:53:00  ||  22876WON
04-06-2020 15:53:24  ||  22876WON
04-06-2020 15:53:25  ||  29805OYF
04-06-2020 15:53:57  ||  29805OYF
04-06-2020 15:53:57  ||  22876WON
04-06-2020 15:54:26  ||  63067FVA
04-06-2020 15:54:26  ||  29805OYF
04-06-2020 15:54:42  ||  10580QVN
04-06-2020 15:54:56  ||  10580QVN
04-06-2020 15:54:59  ||  63067FVA
```

Figure 3: RPI measurements captured at location B and E



Location B

```
04-06-2020 15:40:18  ||  37172RZN
04-06-2020 15:40:39  ||  37172RZN
04-06-2020 15:41:11  ||  37172RZN
04-06-2020 15:41:40  ||  37172RZN
04-06-2020 15:42:38  ||  42026IWJ
04-06-2020 15:43:10  ||  42026IWJ
04-06-2020 15:43:39  ||  42026IWJ
04-06-2020 15:44:21  ||  59043DZP
04-06-2020 15:44:40  ||  39420WCL
04-06-2020 15:44:57  ||  59043DZP
04-06-2020 15:45:13  ||  59043DZP
04-06-2020 15:45:41  ||  59043DZP
04-06-2020 15:45:46  ||  39420WCL
04-06-2020 15:46:11  ||  11466LCF
04-06-2020 15:46:23  ||  39420WCL
04-06-2020 15:46:34  ||  35578PFE
```

User 1

```
04-06-2020 15:40:09  ||  37172RZN
04-06-2020 15:42:09  ||  42026IWJ
04-06-2020 15:44:10  ||  59043DZP
04-06-2020 15:46:10  ||  11466LCF
04-06-2020 15:48:10  ||  51288EFA
04-06-2020 15:50:10  ||  50853OGO
04-06-2020 15:52:10  ||  22876WON
04-06-2020 15:54:10  ||  63067FVA
04-06-2020 15:56:10  ||  59092GVB
04-06-2020 15:58:10  ||  54083BQR
04-06-2020 16:00:10  ||  13947ZGU
04-06-2020 16:02:10  ||  18975MCY
04-06-2020 16:04:10  ||  47066BVU
04-06-2020 16:06:10  ||  56188NBB
```

Location E

```
04-06-2020 15:52:23  ||  60118JSB
04-06-2020 15:52:31  ||  22876WON
04-06-2020 15:52:36  ||  29805OYF
04-06-2020 15:52:56  ||  29805OYF
04-06-2020 15:53:00  ||  22876WON
04-06-2020 15:53:24  ||  22876WON
04-06-2020 15:53:25  ||  29805OYF
04-06-2020 15:53:57  ||  29805OYF
04-06-2020 15:53:57  ||  22876WON
04-06-2020 15:54:26  ||  63067FVA
04-06-2020 15:54:26  ||  29805OYF
04-06-2020 15:54:42  ||  10580QVN
04-06-2020 15:54:56  ||  10580QVN
04-06-2020 15:54:59  ||  63067FVA
```

Figure 4: Profiling User 1 movements

there. Since we know where users are, at which time, and how long they spend at each observed place, it is possible to aggregate relevant information from the users with low effort to potentially de-anonymize them.

A determined attacker can extract significant information about users releasing their Daily Tracing Keys. For example, our experiment simulates the scenario that User 1 lives in the residential area near location A, and may have health and legal problems due to his visits to the clinic and the police station. User 2, on the other hand, might be involved with the municipal administration and seems to like products available in a head shop or at a sports bookmaker. Moreover, since the measurements reveal that both users left location B at about the same time, and, even arrived at the pub (location E) at the same time, spent time there, and also left the pub at the same time, it is likely that these two users may have a social relationship.

We have conducted a series of experiments of different complexity. Our experiments demonstrate the power that the adversary gains by having access to RPI data of individual users. Since the Daily Tracing Keys (DTK) change every 24 hours, traceability across longer time frames would initially not seem to be possible. However, since infected users upload DTKs of 14 days and since typical travel patterns of individual users show marked similarities even between different days (e.g., the typical commute pattern between home and workplace), it is possible to link and track at least some infected users for time periods significantly longer than the validity periods of individual

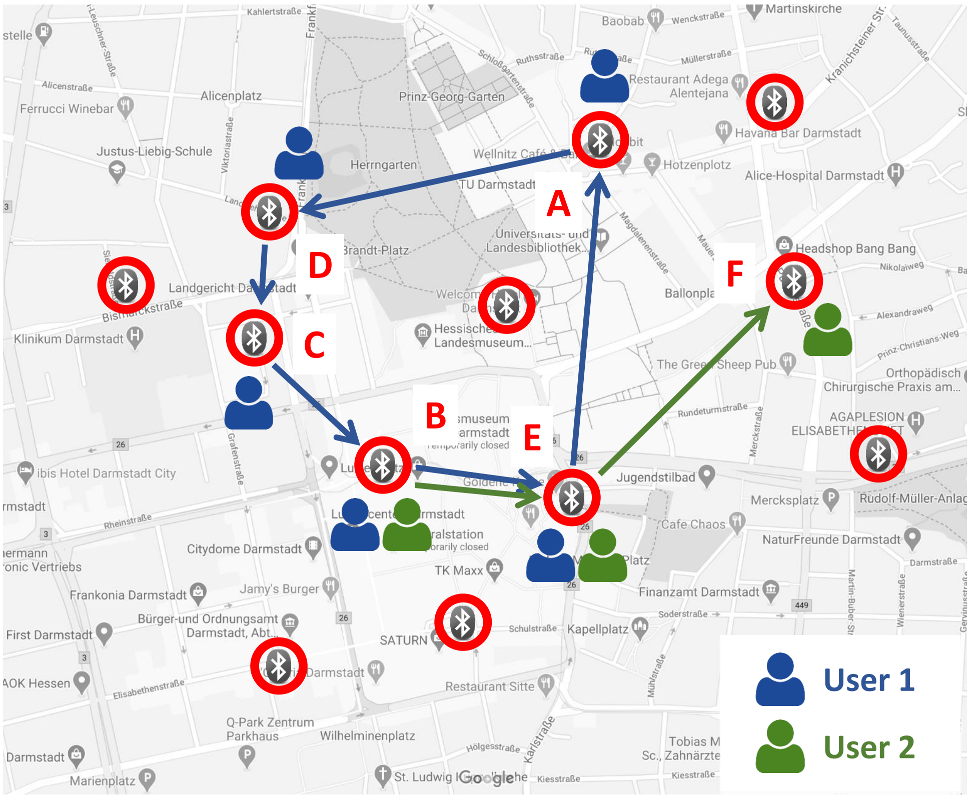


Figure 5: Movement profile of two infected users (User 1 in blue and User 2 in green) based on the observation points.

DTKs (up to 14 days). This clearly will reveal even more personal information and activities of the targeted users and provide ample opportunities for using potentially available additional public information to de-anonymize the users in question. Moreover, de-anonymization becomes easier if the adversary has access to additional information about social relationships of users, e.g., the social graph of an online social network (OSN). This graph can be used to identify the infected users and their social contacts by comparing the social graph of the infected users obtained by the profiling attack to the OSN social graph [16, 25].[5]

4 Mind the Security GAP

The GAP design is vulnerable to relay-based wormhole attacks that can generate fake contacts resulting in a huge number of false alarms about contacts with infected persons, potentially leading to a high pressure on the public healthcare system administering COVID-19 tests. In the worst case, this attack could have the potential of rendering a GAP-based contact tracing system useless. In the rest of this section, we present our real-world attack on the security of the GAP scheme based on a substitute of the GAP in a running app, namely DP-3T's pre-standard *SampleApp* [24]. The GAP is strongly influenced by the DP-3T protocol, but is implemented at the operating system level to allow for more efficient operation as a background task. Thus, DP-3T's prestandard *SampleApp* is "the next best thing" to use, given that currently the official GAP API is only permitted to be used for software development and testing by teams authorized by governmental health institutions.

[5] Interestingly, a similar weakness was observed and criticized [8] for the centralized tracing app, called PEPP-PT [22], that enables the central server to build the social graph of infected individuals!

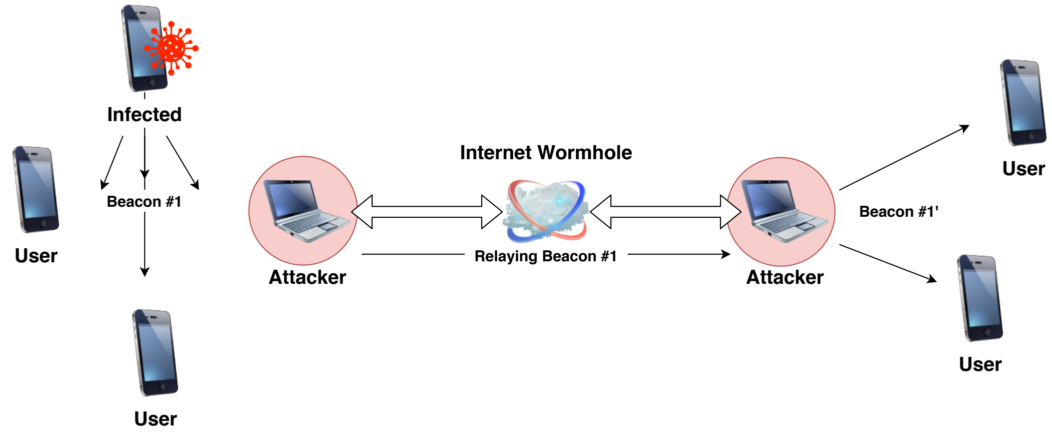


Figure 6: Setup of a wormhole attack to relay COVID-19 tracing beacons

4.1 Security Attack: Infection Alarms Going Viral

A wormhole attack is a particular type of relay attack. Here, an attacker records messages at one physical location of a network, forwards them through a network tunnel to another physical location, and re-transmits them there as if they had been sent at this location in the first place [13].

4.1.1 Goal and System Setup

The goal of the attack is that two or more physical locations are combined into one large logical location. If highly frequented physical locations, such as different train stations, shopping malls etc., are logically linked to each other, one infected person has a stronger effect on other participants in the system, simply due to fact that this infected person has now been (falsely) observed to have been in contact with a significantly increased number of persons. This can lead to a massively higher number of false alarms for people who in reality might not have been in contact with anyone who is infected. If more than two locations are linked together and appear like a single location, then the effect of a COVID-19 positive person on the falsely calculated exposure risk factor of other participants is even stronger. As a result, many more people will want to get tested, since the contact tracing app will suggest that they have been exposed to an infected person. This might in turn exceed the testing capacities available for the citizens, leading to fear and frustration within the general public. Furthermore, people might refuse to use the app, since the false positive rate seems to be too high, i.e., not helpful in contact tracing, but rather causing unnecessary work and more confusion.

Fig. 6 shows the basic setting in which a wormhole attack on contact tracing approaches can be performed. The attacker uses (at least) two Bluetooth LE (BLE) enabled wormhole devices in different locations, each of them in physical proximity to the mobile devices of potential victims, and records the BLE messages broadcast by the users' mobile devices at each location. These recorded BLE messages are then transferred, e.g., via an Internet link, in both directions between the wormhole devices of the attacker and are then re-broadcast to all mobile devices of users in the near vicinity of both locations of the wormhole devices. Thus, these BLE messages seem to come from a local 1-hop neighbor.

This setup allows an attacker to create false positive reports for a potentially large number of persons. An attacker with access to a valid positive COVID-19 diagnosis could transmit his or her personal beacon to multiple locations of interest, like the headquarters of business competitors, the buildings of the national health institute, or the house of parliament, before disclosing his or her *Daily Tracing Keys* to other users as being infectious. In this way, a targeted selection of users can be sent into a false quarantine.

In the GAP approach, the *Rolling Proximity Identifier* is only valid for a known time interval, currently 10 minutes. Based on this knowledge, an attacker performing a wormhole attack can add an expiration date to each received beacon message before transferring it through the Internet link to other wormhole devices. These wormhole devices can then re-broadcast the same message over and over again until the expiration date is met. While the used bandwidth of the Internet wormhole is kept to a minimum, the efficiency of the wormhole devices is maximized, since a single BLE message from a wormhole device can be resent multiple times to victims' devices for as long as the beacon is valid (up to 10 minutes). Due to the fact that Bluetooth beacon messages are relatively small in size and the Internet provides fast communication, relaying is achieved in a matter of seconds.

Such wormhole attacks can be realized easily with a relatively low budget and can be implemented using higher-than-normal signal strengths and/or high-gain antennas to significantly enlarge the area of effect for each wormhole device. Thus, it is possible to create falsely positive contacts between a massive amount of users and amplify the number of persons who will be (falsely) prompted to self-quarantine and seek testing for COVID-19.

4.1.2 Experimental Results

As mentioned earlier, the GAP approach is not permitted to be used by developers or testers that are not members of governmental health institutions, since a special permission of either Google or Apple is required to use their GAP (*Exposure Notification*) API. To validate that a wormhole attack was successful, we need to verify that the Bluetooth LE broadcast beacons sent by a smartphone A to a wormhole device and transmitted over the network to another wormhole device are finally accepted by another smartphone B. This can be done by logging the messages received by smartphone B or presenting status information about BLE communication on the display of smartphone B.

To demonstrate the feasibility of a wormhole attack on the GAP design, the original implementation of DP-3T can be used. Since the GAP approach is "very heavily inspired by the DP-3T group and their approach" [10], DP-3T can be considered as a substitute for GAP to show the success of a wormhole attack. The original DP-3T implementation (later named *prestandard* DP-3T[6]) was available before the GAP API made it into Android and iOS. In this *prestandard* version, DP-3T generates *Daily Tracing Keys*, as well as handles and stores *Rolling Proximity Identifiers* as part of the app. The operating system (iOS or Android) is in charge of handling the lower-level BLE communication, and the app provides the payloads required for BLE communication. In the GAP approach, these components are implemented as part of the operating system. Thus, our experiments were carried out with DP-3T's prestandard *SampleApp* using iOS and Android smartphones, since this app provides messages logs and status information about BLE communication.

For our real-world wormhole attack, we have built a multi-location wormhole for forwarding and rebroadcasting BLE messages using off-the-shelf Raspberry Pis with integrated Bluetooth LE and an Internet uplink. These Raspberry Pis were connected using a central MQTT server as a back-end system for distributing the received BLE messages between each wormhole device. One of these Raspberry Pis functioned as a mobile node by using a battery pack and a mobile phone network uplink.

Our evaluation wormhole connects several physical locations in the cities of Marburg, Gießen, and Darmstadt, Germany. Part of this setup within the city of Marburg is shown in Figure 7. Each Raspberry Pi receives and records all BLE messages sent by surrounding users' iOS and Android smartphones and sends them to the MQTT wormhole server. All other connected wormhole Raspberry Pis receive a copy of these beacons from the MQTT wormhole server and rebroadcast them using their integrated Bluetooth LE hardware. Each wormhole device is sender and receiver at the same time. Thus, this setup works in a bi- or multi-directional fashion.

In our tests, we used several iOS and Android smartphones with the corresponding implementation of the DP-3T prestandard *SampleApp* at multiple physical locations in the three German cities mentioned above. Our tests showed that the DP-3T prestandard *SampleApp* is vulnerable to our wormhole attack. For example, we successfully established a logical contact between smartphones that were 40 kilometers apart in two cities without a real-world contact between the users of these smartphones. In principle, the attack has no distance-related limitations, since the Internet provides sufficiently fast communication to forward Bluetooth LE messages (i.e., RPIs) in a matter of seconds, while the attacker has up to a 10-minute time window. The logical contact between

[6] `https://github.com/DP-3T/dp3t-sdk-ios/releases/tag/prestandard`

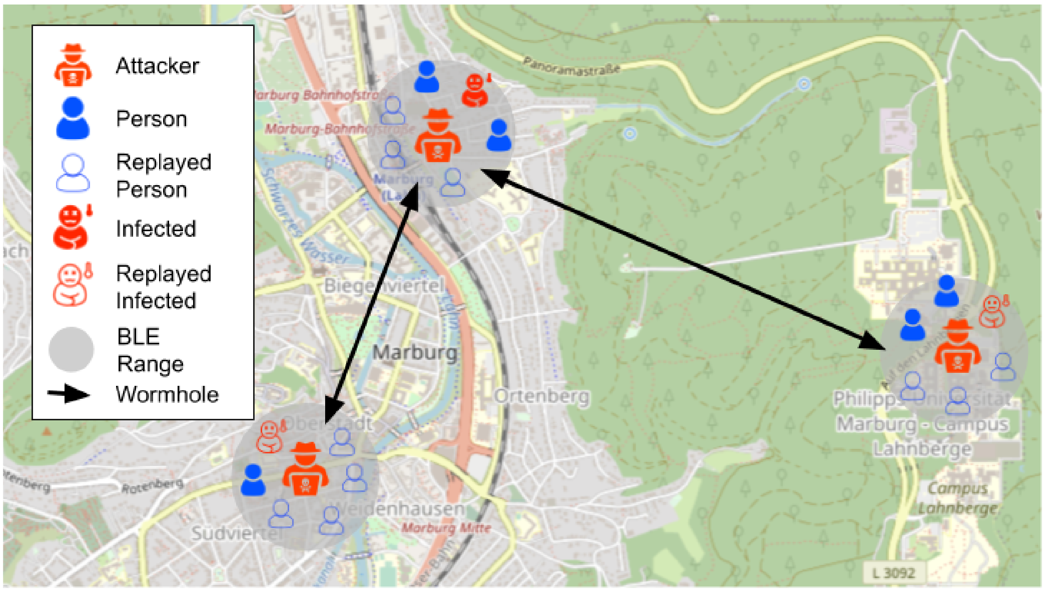


Figure 7: Sketch of the wormhole attack in the city of Marburg

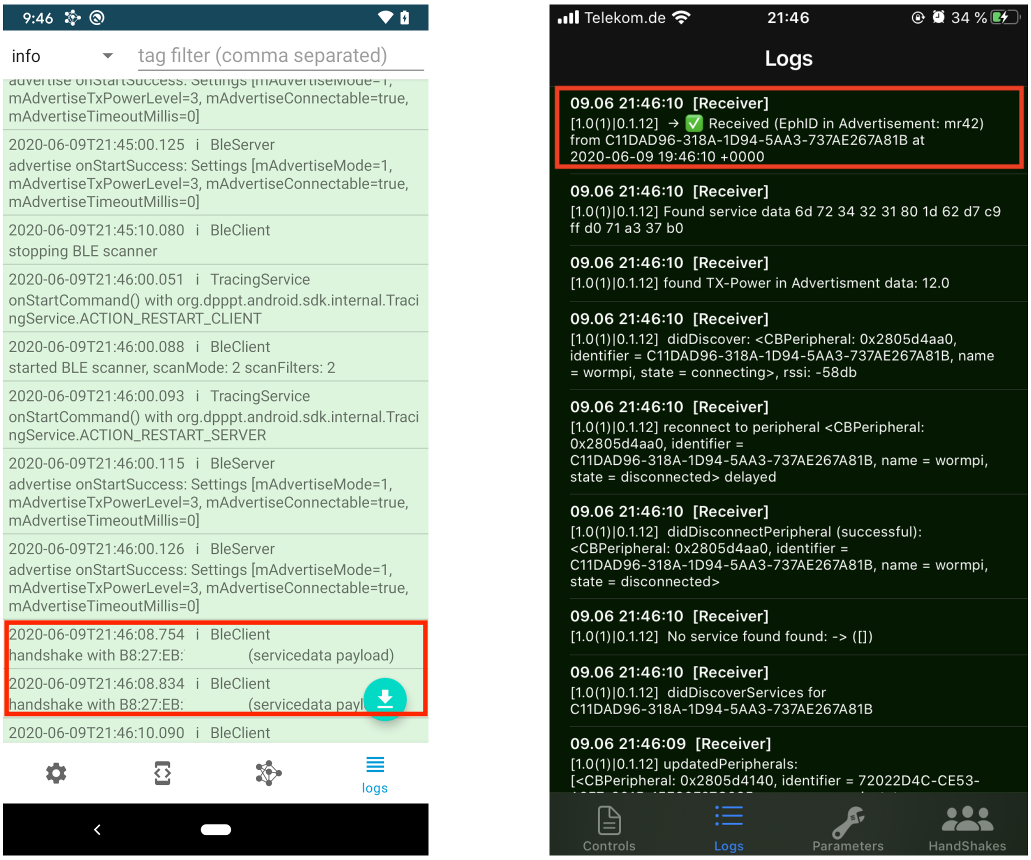
the smartphones was generated without any actions by the users required on their smartphones and without any physical interaction between the two individuals.

```
1  Jun 09 20:45:13 wormpi-mr wormhole[472]: [provider    ] [INFO] [in ] [7E:09:47:A6:EE:7F]
        [Dp3t_ScanRequest] fd68
2  Jun 09 20:45:13 wormpi-mr wormhole[472]: [wormhole-out] [INFO] [7E:09:47:A6:EE:7F]
        [Dp3t_ScanRequest] fd68
3  Jun 09 20:45:13 wormpi-mr wormhole[472]: [wormhole-in ] [INFO] [5A:A2:81:40:7A:B3]
        [Dp3t_ScanResponse] fd68 6d:72:34:32:30:80:1d:62:d7:c9:ff:d0:71:a3:37:b0
4  Jun 09 20:45:13 wormpi-mr wormhole[472]: [provider    ] [INFO] [out] [5A:A2:81:40:7A:B3]
        [Dp3t_ScanResponse] fd68 6d:72:34:32:30:80:1d:62:d7:c9:ff:d0:71:a3:37:b0
```

Listing 1: Raspberry Pi running our wormhole implementation

In Listing 1, an excerpt of the log of a running wormhole device, here called *wormpi-mr*, is shown. The software on the wormhole device consists of a BLE controller and a beacon distribution task, called "provider". The DP-3T prestandard *SampleApp* used the Bluetooth LE UUID `fd68`, which is correctly identified as `Dp3t_ScanRequest` in the case of an empty payload and `Dp3t_ScanResponse` when the RPI is included. In Lines 1-2, another wormhole device has submitted a `ScanRequest` beacon to the provider, indicated by "in", that is then broadcast by *wormpi-mr's* BLE controller ("wormhole-out"). In Lines 3-4, a response is received by *wormpi-mr's* BLE controller as "wormhole-in", which is then sent over the provider ("out") to all other wormhole devices.

In Figure 8, two screenshots of running DP-3T prestandard *SampleApp* instances on Android (Fig. 8a) and iOS (Fig. 8b) are shown. The experiments indicate that in both implementations the execution of a wormhole attack was successful. The Android implementation on a smartphone located in Marburg (Fig. 8a) displays a handshake with the MAC address of the wormhole device (indicated by the rectangles in red), which in this experiment is the hardware MAC address of the used Raspberry Pi (abbreviated due to privacy reasons). The iOS implementation on a smartphone located in Gießen (Fig 8b) is less verbose, but also confirms receiving a beacon with the manually set ephemeral ID of "mr42" (i.e., the smartphone in Marburg; indicated by the rectangle in red), even though the smartphone is not in physical proximity of another smartphone running the DP-3T prestandard *SampleApp*.



```
advertise onStartSuccess: Settings [mAdvertiseMode=1,
mAdvertiseTxPowerLevel=3, mAdvertiseConnectable=true,
mAdvertiseTimeoutMillis=0]
2020-06-09T21:45:00.125   i   BleServer
advertise onStartSuccess: Settings [mAdvertiseMode=1,
mAdvertiseTxPowerLevel=3, mAdvertiseConnectable=true,
mAdvertiseTimeoutMillis=0]
2020-06-09T21:45:10.080   i   BleClient
stopping BLE scanner
2020-06-09T21:46:00.051   i   TracingService
onStartCommand() with org.dpppt.android.sdk.internal.Traci
ngService.ACTION_RESTART_CLIENT
2020-06-09T21:46:00.088   i   BleClient
started BLE scanner, scanMode: 2 scanFilters: 2
2020-06-09T21:46:00.093   i   TracingService
onStartCommand() with org.dpppt.android.sdk.internal.Traci
ngService.ACTION_RESTART_SERVER
2020-06-09T21:46:00.115   i   BleServer
advertise onStartSuccess: Settings [mAdvertiseMode=1,
mAdvertiseTxPowerLevel=3, mAdvertiseConnectable=true,
mAdvertiseTimeoutMillis=0]
2020-06-09T21:46:00.126   i   BleServer
advertise onStartSuccess: Settings [mAdvertiseMode=1,
mAdvertiseTxPowerLevel=3, mAdvertiseConnectable=true,
mAdvertiseTimeoutMillis=0]
2020-06-09T21:46:08.754   i   BleClient
handshake with B8:27:EB:            (servicedata payload)
2020-06-09T21:46:08.834   i   BleClient
handshake with B8:27:EB:            (servicedata payl
2020-06-09T21:46:10.090   i   BleClient
```

(a) Android located in Marburg

```
09.06 21:46:10  [Receiver]
[1.0(1)|0.1.12]  → ✅ Received (EphID in Advertisement: mr42)
from C11DAD96-318A-1D94-5AA3-737AE267A81B at
2020-06-09 19:46:10 +0000

09.06 21:46:10  [Receiver]
[1.0(1)|0.1.12] Found service data 6d 72 34 32 31 80 1d 62 d7 c9
ff d0 71 a3 37 b0

09.06 21:46:10  [Receiver]
[1.0(1)|0.1.12] found TX-Power in Advertisment data: 12.0

09.06 21:46:10  [Receiver]
[1.0(1)|0.1.12]  didDiscover: <CBPeripheral: 0x2805d4aa0,
identifier = C11DAD96-318A-1D94-5AA3-737AE267A81B, name
= wormpi, state = connecting>, rssi: -58db

09.06 21:46:10  [Receiver]
[1.0(1)|0.1.12] reconnect to peripheral <CBPeripheral:
0x2805d4aa0, identifier =
C11DAD96-318A-1D94-5AA3-737AE267A81B, name = wormpi,
state = disconnected> delayed

09.06 21:46:10  [Receiver]
[1.0(1)|0.1.12]  didDisconnectPeripheral (successful):
<CBPeripheral: 0x2805d4aa0, identifier =
C11DAD96-318A-1D94-5AA3-737AE267A81B, name = wormpi,
state = disconnected>

09.06 21:46:10  [Receiver]
[1.0(1)|0.1.12]  No service found found: -> ([])

09.06 21:46:10  [Receiver]
[1.0(1)|0.1.12]  didDiscoverServices for
C11DAD96-318A-1D94-5AA3-737AE267A81B

09.06 21:46:09  [Receiver]
[1.0(1)|0.1.12] updatedPeripherals:
[<CBPeripheral: 0x2805d4140, identifier = 72022D4C-CE53-
```

(b) iOS located in Gießen

Figure 8: DP-3T prestandard *SampleApp* instances with confirmed beacons transmitted through the wormhole "wormpi"

4.1.3 Discussion

If directional high-gain antennas are used at the wormhole devices, the attacker can pinpoint individuals and make them appear at a different physical location through the wormhole. For example, the attacker can use a large number of collected GAP *Rolling Proximity Identifiers* to make a victim appear to have had a high number of contacts, ultimately resulting in a high probability of an infected contact and a restriction of the user's freedom, since he or she must most likely follow the instructions of self-quarantine. In addition, an (unnecessary) medical test for an infection is likely to be performed, which puts strain on the available test capacities. By using directional wormhole devices, it is possible to collect identifiers from (self-)quarantined and possibly infectious persons and broadcast them in highly frequented areas in order to create virtual super-spreading events. This is particularly relevant in regions and during times where test capacities are not sufficiently available. While this has no direct impact on an individual user, it also puts strain on existing testing capacities and distracts the focus of governmental health staff from real cases running in parallel.

Unfortunately, all proposed contact tracing apps are more or less vulnerable to wormhole attacks, unless they apply mechanisms that may affect their privacy-preserving properties or significantly increase their resource demands, particularly in terms of battery consumption on mobile devices, e.g., when GPS is used for evaluating location information. We then have the choice between Scott McNealy's famous statement (1999): "You have zero privacy anyway. Get over it." [27], recent individual opinions in some public polls: "If you can save the lives of people, who cares about privacy?", and undesirably fast battery drains. Nevertheless, we strongly believe that having to give up privacy may lead to living conditions that are not tolerable in the long run for many individuals in any society.

5 Conclusion

We presented experimental results demonstrating that in real-world scenarios the current GAP design is vulnerable to (i) profiling and possibly de-anonymizing infected persons, and (ii) relay-based wormhole attacks that principally can generate fake contacts with the potential of significantly affecting the accuracy of an app-based contact tracing system.

Since currently the GAP API is only permitted to be used by apps authorized by governmental health institutions [14], we could only evaluate a GAP tracing simulator implemented by following the cryptography API specification of GAP and the publicly available DP-3T prestandard *SampleApp* that strongly inspired the design of the GAP API. Since the fundamental vulnerabilities are based on the GAP scheme itself, all GAP-based apps (including the Swiss SwissCovid app and the German Corona-Warn-App) face the same issues. To increase trust in and public adoption of contact tracing apps, it is not only necessary to open source the national contact tracing app codes, but also the corresponding operating system functionality by Google and Apple. This is where the "magic" happens, and this is another part where potential privacy leaks could occur. Thus, for a complete audit and sound statements regarding security and privacy issues of contact tracing apps, it is necessary to have the full software stack (i.e., app code, GAP API code, and backend-server code) available as source code and in executable form on different mobile platforms.

In their current versions, GAP-based contact tracing apps still have room for improvement in terms of privacy and security. Since the primary issues lie within the GAP approach itself, a revision of the GAP protocol might be required. For example, the currently used beacon payload does not offer the required storage space for additional information to implement countermeasures against the presented attacks [15]. However, some of the countermeasures would probably pose additional threats to the privacy and security properties of currently proposed decentralized contact tracing apps or would most likely lead to unacceptably high resource demands, particularly in terms of battery consumption.

References

[1] Ross Anderson. Contact Tracing in the Real World, April 2020. `https://www.lightbluetouchpaper.org/2020/04/12/contact-tracing-in-the-real-world/`.

[2] Apple and Google. Exposure Notification: Cryptography Specification, v1.2, April 2020. `https://www.apple.com/covid19/contacttracing`.

[3] Department of Health Australian Government. CovidSafe Contact Tracing App. `https://www.health.gov.au/resources/apps-and-tools/covidsafe-app`.

[4] Wasilij Beskorovajnov, Felix Dörre, Gunnar Hartung, Alexander Koch, Jörn Müller-Quade, and Thorsten Strufe. ConTra Corona: Contact Tracing against the Coronavirus by Bridging the Centralized–Decentralized Divide for Stronger Privacy. Cryptology ePrint Archive, Report 2020/505, apr 2020. `https://eprint.iacr.org/2020/505`.

[5] Justin Chan, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Sudheesh Singanamalla, Jacob Sunshine, et al. PACT: Privacy-Sensitive Protocols And Mechanisms for Mobile Contact Tracing. *arXiv preprint arXiv:2004.03544*, apr 2020.

[6] Pan-European Privacy-Preserving Proximity Tracing Consortium. Documentation for Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT). `https://github.com/pepp-pt/pepp-pt-documentation`.

[7] Bennett Cyphers and Gennie Gebhart. Apple and Google's COVID-19 Exposure Notification API: Questions and Answers. `https://www.eff.org/deeplinks/2020/04/apple-and-googles-covid-19-exposure-notification-api-questions-and-answers`.

[8] Decentralized Privacy-Preserving Proximity Tracing (DP-3T). Security and Privacy Analysis of the Document "PEPP-PT: Data Protection and Information Security Architecture", 2020. https://github.com/DP-3T/documents.

[9] e-Health Network. Mobile Applications to Support Contact Tracing in the EU's Fight Against COVID-19, 2020.

[10] Darrell Etherington and Natasha Lomas. Apple and Google Update Joint Coronavirus Tracing Tech to Improve User Privacy and Developer Flexibility.

[11] Ministry of Health Government of Singapore. TraceTogether Contact Tracing App. `https://www.tracetogether.gov.sg/`.

[12] Yaron Gvili. Security Analysis of the COVID-19 Contact Tracing Specifications by Apple Inc. and Google Inc. Cryptology ePrint Archive, Report 2020/428, apr 2020. `https://eprint.iacr.org/2020/428`.

[13] Yih-Chun Hu, Adrian Perrig, and David B Johnson. Wormhole Attacks in Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370–380, 2006.

[14] Apple Inc. Exposure Notification Addendum. `https://developer.apple.com/contact/request/download/Exposure_Notification_Addendum.pdf`.

[15] Apple Inc. Exposure Notification Bluetooth Specification v1.2. `https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-BluetoothSpecificationv1.2.pdf`.

[16] Shouling Ji, Weiqing Li, Prateek Mittal, Xin Hu, and Raheem Beyah. SecGraph: A Uniform and Open-source Evaluation System for Graph Data Anonymization and De-anonymization. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 303–318, Washington, D.C., August 2015. USENIX Association.

[17] Krista Merry and Pete Bettinger. Smartphone GPS Accuracy Study in an Urban Environment. *PLOS ONE*, 14(7):1–19, 07 2019.

[18] Markus Miettinen, Thien Duc Nguyen, and Ahmad-Reza Sadeghi. Comparison of Tracing Approaches. `https://tracecorona.net/comparison-of-tracing-approaches/`.

[19] NHSX. NHS COVID-19: The New Contact Tracing App from the NHS. `https://www.ncsc.gov.uk/information/nhs-covid-19-app-explainer`.

[20] Government of France. StopCovid France. `https://www.economie.gouv.fr/stopcovid#`.

[21] Government of India. Aarogya Setu Mobile App. `https://www.mygov.in/aarogya-setu-app/`.

[22] Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT). Data Protection and Security Architecture: Illustrated on the German Implementation, 2020. https://github.com/pepp-pt/pepp-pt-documentation.

[23] Krzysztof Pietrzak. Delayed Authentication: Preventing Replay and Relay Attacks in Private Contact Tracing. Cryptology ePrint Archive, Report 2020/418, apr 2020. `https://eprint.iacr.org/2020/418`.

[24] DP-3T project. Decentralized Privacy-Preserving Proximity Tracing. `https://github.com/DP-3T/documents`.

[25] Laura Radaelli, Piotr Sapiezynski, Florimond Houssiau, Erez Shmueli, and Yves-Alexandre de Montjoye. Quantifying Surveillance in the Networked Age: Node-based Intrusions and Group Privacy. *CoRR*, abs/1803.09007, 2018.

[26] Ramesh Raskar, Isabel Schunemann, Rachel Barbar, Kristen Vilcans, Jim Gray, Praneeth Vepakomma, Suraj Kapa, Andrea Nuzzo, Rajiv Gupta, Alex Berke, Dazza Greenwood, Christian Keegan, Shriank Kanaparti, Robson Beaudry, David Stansbury, Beatriz Botero Arcila, Rishank Kanaparti, Vitor Pamplona, Francesco Benedetti and| Alina Clough, Riddhiman Das, Kaushal Jain Khahlil Louisy, Greg Nadeau, Steve Penrod, Yasaman Rajaee, Abhishek Singh, Greg Storm, and John Werner. Apps Gone Rogue: Maintaining Personal Privacy in an Epidemic. `https://arxiv.org/pdf/2003.08567.pdf`.

[27] Polly Sprenger. Sun on Privacy: 'Get Over It'. https://www.wired.com/1999/01/sun-on-privacy-get-over-it/.

[28] Serge Vaudenay. Analysis of DP-3T. Cryptology ePrint Archive, Report 2020/399, apr 2020. https://eprint.iacr.org/2020/399.

[29] Serge Vaudenay. Centralized or Decentralized? The Contact Tracing Dilemma. Cryptology ePrint Archive, Report 2020/531, may 2020. https://eprint.iacr.org/2020/531.