A Coronavirus Contact Tracing App Replay Attack with Estimated Amplification Factors

Stephen Farrell, Douglas J. Leith
School of Computer Science & Statistics,
Trinity College Dublin, Ireland
19th May 2020

Abstract—COVID-19 tracing apps making use of Bluetooth Low Energy (BLE) emit random-looking identifiers that can later be used to reveal previous proximity with a person who has tested positive. For privacy reasons these identifiers generally cannot be authenticated by recipients. This creates the potential for an already-known replay attack that we describe in further detail here. In such attacks the bad actor’s goal is to create additional false positive proximity warnings, either to disconcert the people receiving erroneous warnings, or to discredit the overall system. While we point out ways in which the attack could be partly mitigated, we conclude that a) preventing the attack could add significant complexity to the overall system and might not be feasible, b) that the impact of the attack increases as more people run the tracing app, and c) that the attack can be targeted against key staff in some scenarios so that targeting even with a small amplification factor may cause noticeable damage. We present a model for that amplification that implies we could see four or more false positives per hour for the real positive tests that result from use of a targeted testing station whilst it is being attacked.

I. INTRODUCTION

There is currently a great deal of interest in the use of mobile apps using Bluetooth Low Energy (BLE) to facilitate COVID-19 contact tracing. This is motivated by the hope that more efficient and scalable contact tracing might allow the lockdown measures currently in place in many countries to be relaxed more quickly [1].

Almost as soon as such contact tracing apps were suggested, concerns about efficacy, privacy and abuse were raised. [2] One obvious attack is to store and later replay the BLE signals used to infer proximity so as to generate false positives in the event that the original emitter of the signals declares themselves COVID-positive. In this report, we describe this attack in more detail, as it might be attempted against currently described BLE-based proposals. We also consider the amplification factors that might be achievable for an attacker in a realistic scenario.

In wireless sensor networks, attacks similar to this one are well known and have been termed “wormhole” attacks[3], but here we prefer to simply refer to the attack more simply as a kind of replay.

This document was produced as part of the Trinity College Dublin funded “Testing Apps for COVID-19 Tracing” (TACT) project.1 In other TACT documents, we have documented privacy issues with the OpenTrace [4] implementation [5] and with the general efficacy of using BLE for distance estimation in challenging radio environments. [6]

II. ATTACK DETAILS

We describe the attack as it might be attempted against an app using the Google/Apple de-centralised API and similar systems. We then consider how similar attacks might be attempted against other systems such as Singapore OpenTrace app or the UK NHSX app. Note that all systems in this extremely rapidly-developing space have been undergoing significant change on a weekly basis, so the descriptions below may quickly be outdated.

A. Apps using the Google/Apple API

The Google/Apple API is described in [7], [8]. Honest users generate and transmit a new “Rolling Proximity Identifier” (RPI) value in a BLE beacon every 10 minutes, at the same time as the “Bluetooth randomized address” (the layer 2 MAC address) is changed. RPI values are derived from a randomly chosen “Temporary Exposure Key” (TEK) that is changed daily. The TEK plus the index number of the 10-minute timeslot is used to generate the set of RPI values used during the day. When a user tests positive for COVID-19, they upload the TEK values corresponding to the set of days when they might have been in contact with others. Each day, all those other users download the set of TEK values that have been uploaded, re-calculate the set of RPI values that would have been associated with those TEK values and check to see if any of the derived RPI values match those that have been seen on this handset. If there is a match, and if that RPI was seen for a sufficiently long duration (a multiple of 5 minutes) then the app can alert the user and suggest suitable actions, such as contacting medical staff followed by self-isolation in the case where the contact duration is considered sufficiently long to justify this action.

A bad actor implements a collector, perhaps using a small computer such as a Raspberry Pi. The collector listens for COVID-19 related BLE beacons, containing the RPIs, and has a backhaul to the Internet, perhaps using GSM/LTE. Collected beacon values are uploaded to a command-and-control server.2 Note that the collector does not have to be in proximity with an honest handset for the 15 minutes required to constitute a real contact - the collector only needs to care about unique beacon values, so seeing one single instance is enough for its purposes. Those are transmitted roughly every 250ms.

2As beacon values are short (less than 32 octets) a dedicated command-and-control server may not be required and any publicly available service that allows for upload of user generated content could be used, with beacon values steganographically embedded in images or simply as base64 or otherwise encoded text.

1https://down.dsg.cs.tcd.ie/tact/
The bad actor also implements a **spreader**, likely on a similar platform, that periodically downloads new beacon values uploaded by the collector. The spreader re-transmits these beacon values as fast as it can, for the longest duration for which the beacon could possibly be considered “valid” if a positive test has occurred and a TEK value producing that RPI has been uploaded. This duration, during which the spreader re-transmits RPIs will vary by scheme. For the Google/Apple scheme it will be at least 2 hours due to the built-in “+/- 2 hours” tolerance factor. (Note that implementations might well be buggy, in that they might omit a check on the timeliness of a published value and only match on the beacon octets. Such implementations could be vulnerable to this attack for approximately 14 days.)

The effect of the attack is that when a person tests positive and uploads values corresponding to a beacon re-transmitted by the spreader, then everyone who has been in relative proximity to the spreader while it was re-transmitting that beacon for a sufficient duration will produce a false positive. This will certainly at least disconcert the person involved. If the spreader has reached enough people, often enough, then that creates the amplification factor in the attack - instead of just the collector having seen one single beacon value, everyone in proximity with the spreader at the wrong time is treated as a contact. One or more spreaders can be deployed in places where the spreader is likely to be in contact with key workers. So this attack has an amplification factor and can be somewhat targeted.

The bad actor can deploy collectors at sites with a high probability of seeing beacons from an person who has, or is likely to, test positive. For example at a COVID-19 testing station.

The bad actor can deploy spreaders at sites where people are expected to linger and not be COVID-19 positive. For example at the emergency department (ED) waiting room on the non-COVID-19 “side” of a hospital.

The bad actor could use a higher power transmitter for the spreader to increase its effective range. The bad actor could use a larger antenna for the collector to increase its range.

An implementation could have spreaders that also collect. With such an implementation the ED waiting room might be an attractive deployment, as it has likely-uninfected lingerers, but is also likely to see some people who later test positive (perhaps medical staff).

### B. OpenTrace-like

OpenTrace-like schemes, such as have been deployed in Singapore or Australia, have centrally generated, time-based, beacon values called tempIDs. The tempID is a cryptographic data structure [9] containing an authentication tag, an initialisation vector and a ciphertext. The plaintext used is the catenation of a user-ID and the start and end times of the tempID. The key used to encrypt the tempIDs is known to the server-side as this is a centralised solution.

Each tempID is valid for 15 minutes, and batches are downloaded periodically from a central server. When a person tests positive the model is that they then upload the set of tempIDs their phone has seen and trained contract tracers use that information to telephone potential contacts. This is possible as the plaintext form of the tempID contains an identifier for the sender that is linked to a telephone number given to the app at registration time.

The spreader/collector architecture for the attack still works but in an inverted form - the bad actor needs the collector to collect as many tempIDs as possible and to then send those repeatedly to a phone that is likely be carried by someone who later tests positive. In effect, the bad actor swaps the deployments of collectors and spreaders to aim for best amplification.

When the tempIDs are uploaded to the central server following a positive test, the times at which each of those were received are also uploaded. This limits the spreader’s opportunity to “usefully” re-transmit tempIDs to 15 minutes plus whatever time skew is accepted on upload.

As indicated above, a bad actor might choose to deploy a collector here in the non-COVID ED waiting area, and spreaders near COVID test stations.

### C. NHSX-like

For our purposes, the UK NHSX app [10] has the same data flows as OpenTrace, and hence is vulnerable to the same collector/spreader deployment.

In this protocol the value received from the spreader will contain a tuple of a so-called “BroadcastValue” (BV), a time, transmit-power and a symmetric (HMAC-SHA256) message authentication code. The BV is encrypted for the server using public key cryptography, contains as plaintext the identifier of the original sending installation, and is used for 24 hours. The symmetric message authentication code is created with a key shared between the central server and the original sending client. The BLE protocol details also differ in that the connection from the spreader to potential victim is a longer-lived connection, but crucially the message authentication code is not verifiable in real time at the receiver. This means that the amplification factor is determined by two factors: the clock-skew the receiver is willing to accept (not defined in the protocol that we can see) and the length of connection the spreader can maintain with the potential victim.

We note that [10] almost describes the above attack if one combines the “mass notification problem” and “targeted false alert problems” which is essentially the attack we describe here. (Again, we make no claim that this is a new attack - the AODV wormhole attacks from a decade ago are morally equivalent.)

### III. Possible Mitigations

In this section we consider mitigations for the attack against handsets using the Google/Apple API.

The fact that the collector only needs to see one beacon and doesn’t need any extended proximity with the sender is a weakness of the Google/Apple scheme that could be
addressed. For example, inspired by, but different to, “Timed Efficient Stream Loss-Tolerant Authentication” (TESLA), [11] senders could emit $E(k,RPI)$ being the cleartext RPI encrypted under some random key $k$ known only to the sender. Then every $N$-th beacon the sender could emit the key $k$ (perhaps a few times) at which point the receiver can store the RPI and RSSI values. The benefit is that the sender would have to be in proximity with the collector for on average $N/2$ beacons which will reduce the potential amplification factor. The value $N$ might be chosen to be 5 minutes, which corresponds to the minimum duration an app can query from the API. This would however represent a protocol change as well as an implementation/deployment change.

An easier mitigation would be reducing the “+/− 2 hour” acceptable time window which should reduce the amplification factor for this attack substantially. It is unclear why this 4-hour window was chosen.

Implementations should also be tested to ensure that they do actually correctly enforce any time limits that are assumed to be enforced.

Another defence against this attack might be to try detect spreaders in sensitive locations, by using beacon scanning apps that should detect beacons being re-transmitted when there are no known devices in the area. That should work well for areas with a closing/cleaning time, where a security officer could do the scanning, but would not work for an ED waiting area that is open 24×7.

Similarly, one might attempt to “sanity check” the various signal strengths (RSSI and TxPower) to detect the existence of a spreader. It seems that, absent real-time authentication the sender can mimic anything a real sender would do, hence making the values appear “sane.”

One might consider deploying our own “good” collectors, uploading all values seen, and where they were seen, to a central database, and then checking for beacon re-transmissions with unexpected delays or from distant locations, thus detecting the ongoing attack. Sadly, this defence would end up partly-centralising what would otherwise be a de-centralised scheme.

With the Google/Apple API anyone who uploads TEKs would be revealing their proximity to locations near “good” collectors and the times to the owner of that database. If “good” collectors were also deployed with sufficient density, so that beacon values don’t change as a person moves between “good” collectors, then the database might also expose movement patterns in a recoverable manner. (Basically, if we had a database of many beacons, we might be able to infer lots from that.)

If a bad actor succeeds in this attack, it may be that a device, on downloading today’s TEKs, appears to have been in proximity with a larger than expected number of people who tested positive, due to the amplification factor. One might therefore consider not warning the user or treating that situation differently from what is presumably the expected case of one contact generating a warning. However, having been in contact with more than one person who tested positive may not be a rare scenario, given that infections occur in clusters, so it would seem unwise to not warn the user just for this reason.

There is a hard trade off here. If beacons can be authenticated, then they are more identifying. If they cannot, replays are possible. Automotive manufacturers have faced the same issue with the “Basic Safety Message” intended to be broadcast from each vehicle so as to trigger braking warnings and ultimately autonomous vehicle actions such as braking, in dangerous situations. [12] The Public Key Infrastructure (PKI) solution chosen there as a part of IEEE 1609.3, involves setting up an extremely complex and non-standard PKI to authenticate vehicles, so that each vehicle interacts with the PKI roughly annually at “service-time”, but where each vehicle is given a large set of private keys and public key certificates so that a new certificate (and hence less correlatable authentication) can be used every day. In the authors’ view, it seems unclear that that level of complexity is acceptable for this application.

IV. RISK

With the expectation that apps using the Google/Apple API will be widely deployed, the probability of an attack like this being attempted or demonstrated as a proof-of-concept (PoC) seems high. If the PoC code were publicly released, as is common, the probability of copy-cat attack attempts would also seem high.

In that event, if the amplification factor is high, a bad actor could cause many false positive self-isolation events, causing much concern. Even if the amplification factor is low, and only a few false positive events are caused, the bad actor might reveal the fact that the attack had succeeded, in order to reduce confidence in the operation of the overall system, causing people to turn off or not install the app, even in regions where the attack was not attempted.

It therefore seems prudent to consider what kinds of amplification factor may be realised by our bad actor.

V. AN AMPLIFICATION MODEL

In order to provide a way to illustrate what might be a realistic amplification factor for this replay attack, we present a hopefully realistic model in Table I. Our argument here is in two parts: first, do we expect a collector to see more than one beacon per hour that is later associated with a positive test? Secondly, if a spreader has one beacon per hour to transmit that is later associated with a positive test, then what is the expected amplification factor?

If the answer to our first question is yes, and if the answer to the second question is greater than zero, then we will have established a rough minimum value for the amplification factor. (We choose this not-very precise form of argument both because it is simpler to comprehend and because we can’t be very confident of the precision of the numbers used with our model.)

We assume the collector is deployed at a COVID-19 testing site and the spreader in the non-COVID-19 ED waiting room of a hospital as described earlier.

3The actual scenario and so-called “butterfly” cryptographic keying are even more involved and complex but needn’t bother us for now.
TABLE I
MODEL PARAMETERS AND VALUES FOR TEST-CENTRE/ED SCENARIO

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>is the lifetime of a unique beacon (10 mins)</td>
</tr>
<tr>
<td>S</td>
<td>is the accepted clock skew (120 minutes)</td>
</tr>
<tr>
<td>C</td>
<td>is the duration that constitutes a contact (15 mins)</td>
</tr>
<tr>
<td>D</td>
<td>is the deployment percentage for the app (20%)</td>
</tr>
<tr>
<td>B</td>
<td>is the average number of people whose beacons are seen by the collector per hour (55)</td>
</tr>
<tr>
<td>P</td>
<td>is the percentage of those beacons from someone who later tests positive and uploads (3%)</td>
</tr>
<tr>
<td>N</td>
<td>is the average number of people waiting in the ED waiting area for more than C minutes (20)</td>
</tr>
<tr>
<td>W</td>
<td>is the average duration of waiting in the ED waiting area (60 minutes)</td>
</tr>
<tr>
<td>R</td>
<td>is the duration for which the spreader re-transmits each beacon (120 minutes)</td>
</tr>
</tbody>
</table>

One COVID-19 community testing centre in Dublin is reported to involve 30 staff and is expected to do 200-400 tests per day. 4 If that centre is open for 8 hours then that will mean about B = 55 people emitting beacons each hour. We can roughly justify N = 20 based on figures for a local hospital emergency department. 5 In 2018 82,350 patients were treated in that ED. If those people’s attendances were uniformly distributed (they are not!) that would map to 18 people arriving every two hours. Some hospital and ambulance staff will also be present relatively often adding to the count. We therefore assume an average of 20 people at this point due to the lack of concrete data. We use P = 3% based on recent news reporting in Ireland. 6.

The collector should therefore see on average B*P*P*(60/L) unique beacons that map to a positive test each hour. Using our numbers that is 9.9 which is greater than one. The spreader will re-transmit beacons as fast as possible. With the above numbers, at any given time, the spreader will be re-transmitting at least one beacon that will later be uploaded as a result of a positive test. In that case, anyone running the app and within range of the spreader for at least C minutes will be notified that they are a contact at some point. Given C is less than W and R is less than or equal to S, we will see N*D people who are contacted in an hour, despite being false positives.

Based on this we could define the amplification factor in two ways - as the number of false positives caused per spreader, per hour: AmpS; or as the number of false positives caused per true positive, per spreader: AmpP. If either is greater than zero, then the attack succeeds. With the numbers above, that means AmpS=4 false positives per spreader deployed per hour, and AmpP=0.4 with one spreader deployed. However, as D increases, both of our (underestimated) measures of amplification get worse.

VI. RESPONSIBLE DISCLOSURE
Prior to publication, we informed personal contacts in organisations developing COVID-19 tracing apps and APIs. All contacted were aware of the potential for this kind of attack and are considering mitigations in their work. We would like to thank them for their feedback. We consider publication now to be useful so those developing COVID-19 tracing systems are aware of the issue and can consider it in their development and operations.

VII. SUMMARY AND CONCLUSIONS
We describe a replay attack against COVID-19 tracing apps that may allow a bad actor to create false positives for any such app, regardless of whether it is centralised or de-centralised. While the amplification factors may not seem that high, attacks only ever get worse. The fact that the attack is possible may disincent users from installing or turning on such apps.

REFERENCES