SECURITY ANALYSIS OF THE COVID-19 CONTACT TRACING SPECIFICATIONS BY APPLE INC. AND GOOGLE INC.

A PREPRINT

Yaron Gvili
Cryptomnium LLC
cryptomniumllc@gmail.com

April 14, 2020

ABSTRACT

In a joint effort to fight the COVID-19 pandemic, Apple Inc. and Google Inc. recently partnered to develop a contact tracing technology to help governments and health agencies reduce the spread of the virus, with user privacy and security central to the design. The partnership announcement included technical specifications of the planned technology, which has great potential for widespread adoption due to the global reach of the two companies. In this report, we provide a security analysis of these specifications. We show that the current specifications may introduce significant risks to society and propose mitigation strategies for these risks that do not require major changes to the technology and are easy to adopt. Our analysis focuses mostly on system security considerations yet also includes information security considerations. We leave out of scope a discussion on how important or effective the technology is in fighting the pandemic.

***K*eywords** COVID-19 · Contact Tracing · System Security · Information Security

1 Introduction

The COVID-19 pandemic, spread by the SARS-CoV-2 virus, has transformed the world over the last few months. It has forced strong confinement measures in most countries in the world that drastically changed the global economy. The resulting global economic impact is yet to be determined but it is likely this pandemic is one of the worst economic events from a historical perspective.

As part of the response to the pandemic, governments introduced technology tools to monitor the spread of the virus in order to better control and reduce it. These tools aim to introduce automation to epidemiological investigations and to help in limiting the confinement measures to small areas instead of entire cities or countries. However, these tools often rely on tracking the locations of virtually all citizens over a significant period of time, in violation of individual privacy rights. Consequently, researchers proposed solutions for contact tracing based on proximity of mobile devices of individuals while preserving their privacy. We describe some of these propositions in Section 1.1.

Recently, a notable privacy-preserving contact tracing initiative was launched by Apple Inc. and Google Inc. The two companies announced a partnership to build an interoperable COVID-19 contact tracing technology into their mobile platforms [1, 2]. They also published specifications for the technology [3, 4, 5]. This is a promising initiative that has the potential to succeed at a global scale due to the global reach of the two companies as well as their ability to build the technology directly into the operating system layer of their platform for better security.

While the specifications proposed by Apple Inc. and Google Inc. have desirable privacy and security properties, they also open the door to various types of attacks that could pose significant risks to society. In this report, we analyze the specifications, focusing mostly on system security consideration yet also include information security considerations. It is our hope that our analysis will help improve the security of the future system described by the specifications.

1.1 Related Work

Vaudenay [6] provided a security analysis of a similar proposition, the Decentralized Privacy-Preserving Proximity Tracing (DP-3T) system proposed by the Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT) project.

An international group of researchers proposed the Private Automated Contact Tracing (PACT) protocol [8] in an effort to see technical convergence of various similar propositions:

- The Covid-Watch proposal [9] and the related TCN protocol [10].
- The Canetti et al. protocol [11].
- The DP-3T protocol [12].
- A different protocol that goes by the same PACT acronym [13].

The specifications by Apple Inc. and Google Inc. that we consider in this report are similar to the above proposals.

1.2 Our Contribution

Our analysis in this report focuses mostly on systems security (Section 3). This aspect of security has received relatively less attention in related work. The propositions cited in Section 1.1 emphasize information security and we refer to them for more details beyond our information security analysis (Section 4).

Our analysis targets the particular specifications by Apple Inc. and Google Inc. We identify specific risks, consequences and mitigation strategies relevant to these specifications by considering both systems and information security.

2 Overview of the Specifications

The main scenario described in the specifications considers two users, Alice and Bob, each having a mobile device that runs the specified system [3, 4, 5]. Each mobile device uses Bluetooth Low Energy (BLE) to exchange (i.e. advertise and receive) anonymous identifier beacons with other unknown mobile devices in proximity. The generated anonymous identifier, called rolling proximity identifier, changes about every 15 minutes. It is derived from a private daily tracing key, which changes daily, that in turn is derived from a private tracing key which is generated once per device. The mobile device keeps for 14 days (or any period determined by public health authorities) any anonymous identifiers received during this period.

In the scenario, Alice and Bob meet and have a 10-minute conversation. During this time, Alice's mobile device collects anonymous identifiers from Bob's proximate mobile device and vise versa, and possibly other anonymous identifiers from other mobile devices that happen to have been proximate at the time. This is performed by a Contact Detection Service in the mobile device. Later on, Bob is positively diagnosed for COVID-19 and reports this result to a public health authority. With Bob's consent, his mobile device uploads to the cloud the daily tracing keys used to generate his anonymous identifiers in the last 14 days. The uploaded keys are called diagnosis keys. Alice's mobile device periodically downloads the diagnosis keys uploaded to the cloud by anyone in the last 14 days, without knowing who the keys belong to, and finds a match between some of the identifiers it collected in the last 14 days and those derived from a diagnosis key generated in the last 14 days. Alice's mobile device raises an alert of this find to Alice.

The Bluetooth specification [4] adds a few definitions of interest. The Bluetooth configuration is defined to prevent connections to the mobile device and to use a random address. The rolling proximity identifier is changed synchronously with this address so as to prevent linking of rolling proximity identifiers. Scanned advertisements, i.e. events received from the Contact Detection Service, are timestamped but not located. The scanning interval and window may be opportunistic and must be sufficient for discovering a nearby advertisements, i.e. events sent from nearby Contact Detection Services, within 5 minutes.

The cryptography specification [5] adds the following definitions of interest. The external functions used are:

- $\mathrm{HKDF}(\mathrm{Key}, \mathrm{Salt}, \mathrm{Info}, \mathrm{OutputLength})$ designates a specific key derivation function.
- $\mathrm{HMAC}(\mathrm{Key}, \mathrm{Data})$ designates a specific HMAC function.
- $\mathrm{Truncate}(\mathrm{Data}, \mathrm{L})$ designates a function that returns the first L bytes of Data.
- $\mathrm{DayNumber}$ designates the number of 24-hour periods that passed since Unix Epoch Time.
- $\mathrm{TimeIntervalNumber}$ designates the number of 10-minute periods that passed since a 24-hour period started.
- $\mathrm{CRNG}(\mathrm{L})$ designates a function that returns the next L bytes from a cryptographic random number generator.

The key schedule for contact tracing is defined as follows:

- The notation $||$ stands for concatenation.
- The tracing key is defined as $\mathrm{tk} \leftarrow \mathrm{CRNG}(32)$.
- The daily tracing key is defined as $\mathrm{dtk}_i \leftarrow \mathrm{HKDF}(\mathrm{tk}, \mathrm{null}, (\mathrm{UTF8}(\text{"CT-DTK"})||\mathrm{D_i}), 16)$ for day number D_i.
- The rolling proximity identifier for time interval number $\mathrm{TIN_j}$ in day number D_i is defined as $\mathrm{RPI_{i,j}} \leftarrow \mathrm{Truncate}(\mathrm{HMAC}(\mathrm{dtk_i}, (\mathrm{UTF8}(\text{"CT-RPI"})||\mathrm{TIN_j})), 16)$.

3 Systems Security Analysis

This section provides a systems security analysis of the specifications by considering attacks, their consequences, and mitigation strategies for them. In some of the mitigation strategies, we rely on encryption using which a private key assumed to be securely managed at the platform level, like the tracing key is.

3.1 Power and Storage Drain Attacks

Power and storage drain are denial-of-service attacks involving a large volume of messages. While the Bluetooth specification [4] mentions the requirement to deal with large volumes of advertisers, it does not specify how.

Attack: The adversary attacks a nearby device by advertising a large volume of messages from a source. The device is forced to wake up to process each such message. If the message is invalid, the device discards it but has paid a power cost; this is a power drain attack. If the message is valid, the device stores the proximity identifier contained in the message; this is a storage drain attack. A possible extra effect of these attacks may be keeping the attacked device busy, slowing down or preventing its own advertising or other useful work. The attacks are especially efficient in a crowded area where each message impacts many mobile devices. Moreover, the attacker can arbitrarily vary the signal strength of the messages in order to make it hard to determine they are from a common source.

The power and storage drain attacks are briefly mentioned by Vaudenay [6], along with a note that protection measures against denial-of-service attacks exist, however without specifying these measures. In fact, it is not straightforward to counter these attacks. The standard mitigation strategy of recognizing a misbehaving party and banning them by their networking address is infeasible here because of anonymity (e.g. random addresses). For the same reason, the device cannot recognize a common source of multiple valid but rogue messages. Moreover, the power cost is incurred as soon as the mobile device is woken up to process a message.

The Bluetooth specification [4] briefly discusses scan power considerations. It specifies that whenever possible the hardware or operating system should filter out duplicate messages to prevent excessive power drain. However, this strategy does not prevent the power drain attack describe here because most, if not all, of the garbage messages are distinct. Moreover, this strategy clearly does not prevent the storage drain attack.

Consequences: At first, the power drain attack does not appear to be a serious threat because it does not directly impact the services provided by the system. The same is true for the storage drain attack, provided the storage space has not been depleted. However, a user who notices the power drain may decide to opt out of using the system. In particular, this is the case for on-the-go users as well as for users who are sensitive to the cost of recharging their mobile device, who tend to be from a low socioeconomic background. Similarly, a user noticing a storage drain my choose to opt out. If many users would opt out of the system, its effectiveness would be significantly diminished; this is a serious threat.

Mitigation: We propose a number of non-mutually-exclusive mitigation strategies:

1. The system detects the attack and reports it anonymously to the cloud with information on the timing and signal strength of the messages received during the attack. The cloud would receive reports in real-time from all attacked devices. The more reports the cloud has on a particular attack the more accurately it can determine the location of the attacker so it can take appropriate action.
2. The system detects the attack and alerts the user, advising of appropriate behaviors, such as leaving the area and reporting the incident.
3. The system requires that a valid message include a proof-of-work that can be checked by the device hardware without waking up the device. This allows the attacked device to filter out garbage messages at a minimal power cost while imposing a high computational cost sufficient to deter an attacker who advertises a large number of valid but rogue messages.

3.2 Relay and Replay Attacks

In a relay or replay attack, the attacker advertises a message it received from an honest user. In a relay (resp. replay) attack, the message is advertised immediately (resp. significantly) after receiving it.

Attack: In a relay attack, the attacker uses two devices at two disparate locations and advertises from one device any message relayed from the other. In a replay attack, the attacker uses one or two devices at different times and advertises at the later time a message recorded at the earlier time. In a related replay-of-released-cases attack, the attacker advertises a proximity identifier derived from a diagnosis key before the victim observes the publication of this key. The victim of any of these attacks, who is nearby the advertising device of the attacker, receives a proximity identifier with a fake location or time. If one of the proximity identifiers so received is later matched to a diagnosis key, the victim is falsely led to believe they have been in contact with a diagnosed person due to this match.

The relay and replay attacks are analyzed in the PACT protocol paper [8] and other proposals it builds on. The paper offers an effective mitigation strategy that involves storing the timestamp of a received proximity identifier and later matching the timestamp to the time interval number found for the proximity identifier (as discussed below). However, the specifications only include the storing of the timestamp but do not use it in the matching procedure, and so remain exposed to these attacks. Moreover, the PACT protocol and related proposals do not include the stored location for the received proximity identifier in the matching procedure, which allows the relay attack. We therefore believe it is worthwhile discussing these attacks and their mitigation strategies here in more detail.

The relay and replay attacks are also analyzed by Vaudenay [6] and are proposed mitigation strategies for. However, these mitigation strategies require bidirectional communication between the advertising and receiving devices in order to implement a challenge-response protocol. This has the disadvantage of qualitatively changing the communication architecture of the system, leading to various other issues, such as a more complex system that consumes more power and is exposed to accidental or rogue communication drops. We show below *mitigation strategies that do not require changing the communication architecture of the system* and are easy to adopt.

Consequences: If the relay or replay attacks are allowed to be executed at scale, many victims will falsely believe they have been in contact with a diagnosed person. Besides the anxiety of the victims and their close ones, the victims will go get diagnosed for no good reason. This will not only waste precious diagnosis resources but could also lead to a loss of trust in either the system, the diagnosis procedure, or both.

Mitigation: We first consider the relay and replay attacks. The system is modified to require that the Data argument to the HMAC function in the construction of the rolling proximity identifier also include the current location cell number $\mathrm{LCN_k}$ of the advertising device:

$$\mathrm{RPI_{i,j,k}} \leftarrow \mathrm{Truncate}(\mathrm{HMAC}(\mathrm{dtk_i}, (\mathrm{UTF8}(\text{"CT-RPI"})||\mathrm{TIN_j}||\mathrm{LCN_k})), 16)$$

where the possible location cells are predefined in the system and form a partitioning of the space. We note that coarse-grained self-locating is sufficient and can be done efficiently; we defer a discussion on methods for self-locating to Appendix A. Now, the modification limits the scope of the attack to within the location cell. Moreover, it has no bearing on privacy, since a user who receives such a proximity identifier already knows his or her current location (to some granularity), just like they know the current time. The device associates a private encryption of the location cell with the received proximity identifier it stores. The matching procedure is modified to generate from a diagnosis key a larger set of proximity identifiers, by enumerating not only all possible time interval numbers but also all possible location cells, and accept a matching identifier if it is also associated with the location cell used to generate it. It may be desirable to optimize this procedure by limiting the enumerated set of location cells given some (privately encrypted) extra information, e.g. the cities visited during the day in question[1]. We expect that the increased computational work required by the modified matching procedure would not be a problem since it is not invoked frequently.

We now consider a mitigation strategy for the replay-of-released-cases attack. The device associates a private encryption of the current day number and current time interval number with the received proximity identifier it stores. The cloud publishes the diagnosis key in the beginning of the next time interval number after it is reported along with the publishing time t. The matching procedure ignores any proximity identifier associated with a time (calculated from the associated day number and time interval number) at or beyond t.

To further strengthen security, the matching procedure is modified to accept a matching identifier only if it is also associated with the day number and time interval number used to generate it. This modification limits the scope of the attack to within the time period defined by the day number and time interval number.

Finally, we note that the availability of time and location information does not adversely impact security because an adversary already has this information given the protocol defined in the specifications.

[1]For privacy, the user may prefer such information not be automatically sensed and stored, opting to input it to the procedure.

3.3 Trolling Attacks

Trolling attacks are intended to cause false positives, i.e. to falsely lead people to believe they have been in proximity of a diagnosed person. We discussed other types of attacks that can cause false positives in Section 3.2.

Vaudenay [6] discusses trolling attacks as part of an analysis of false alert injection attacks, but does not discuss the particular attack described here. Anderson [7] briefly discusses the attack we consider here but does not offer a mitigation strategy for it.

Attack: The adversary is a person who is diagnosed or expects to be diagnosed soon, and is interested in causing unsuspecting people to fear they have been exposed to the virus. The adversary attaches his mobile device to a carrier, such as a dog or a car or a drone, that runs around and advertises proximity identifiers to the devices of the unsuspecting people. The adversary may also choose to target specific unsuspecting persons.

Consequences: The unsuspecting people will fear they have been exposed to the virus and will go get diagnosed for no good reason. If done at sufficient scale, this attack could not only waste precious diagnosis resources but could also lead to a loss of trust (at least locally) in either the system, the diagnosis procedure, or both.

Mitigation: When unsuspecting people approach public health authorities for diagnosis, the system allows the public health authorities to gather the time and location (collected as described in Section 3.2) of proximity identifiers collected by the unsuspecting people's mobile devices that match the adversary's diagnosis key. While this has an impact on privacy, we note that public health authorities already conduct epidemiological investigations that gather this information manually. If the attack is done at sufficient scale, the public health authorities will notice a concentration of matches due to a specific diagnosis key, at various times and locations that would appear as an abnormal trace. Thus, the public health authorities will recognize the attack as well as the attacker and will take appropriate action. The system also allows the public health authorities to securely publish (e.g. using digital signatures and authentication) revocations of times and locations for a diagnosis key that would be ignored by the matching procedure. If the attack is targeted at specific unsuspecting persons, it may still produce an abnormal trace (e.g. a drone's trace would look very different from a person's) or a trace that appears to deliberately target the unsuspecting persons.

4 Information Security Analysis

While this is not explicit in the specifications, we assume that the channel between the mobile device and the cloud is authenticated. This is also recommended by Vaudenay [6]. Section 3 discussed mitigation strategies for attacks that exploit the non-authenticated channel between mobile devices. We also assume that false reports to public health authorities, discussed by Vaudenay [6], are mitigated, e.g. using permission numbers as discussed by Rivest et al. [8].

We assume that the system, being implemented at the operating system level, does not allow custom applications to access the information it gathers except through interfaces to very specific use cases. This security measure mitigates the risk of rogue applications building a database of proximity events that can be enriched with external information and analyzed. It also mitigates the risk of coercion, in which the attacker (possibly physically) forces targeted persons to disclose proximity events collected in their mobile devices. Vaudenay [6] discussed these risks (as nerd and coercion attacks) and concluded that a Trusted Platform Module (TPM) is required. Our view is that operating-system level security is sufficient mitigation for these risks, given that the operating system is trusted.

4.1 Tracking and Deanonymization Attacks

Tracking and deanonymization attacks are intended to violate privacy. Vaudenay [6] considered such attacks, in one case relying on false alert injections, and offered other mitigation strategies, in some cases relying on a TPM. Our analysis in Section 3 showed how to mitigate false alert injections, and we have argued that a TPM is not required. In this section, we present additional mitigation strategies to tracking and deanonymization attacks.

Attack: The attacker is interested in exposing the identity of a person, or several persons in a meeting, whose mobile device advertised messages including particular proximity identifiers. To this end, the attacker tracks the Bluetooth signal strength of the messages as well as the location of the attacker's mobile device when each message was received, and uses this information to infer locations over time of the targeted person's mobile device to good accuracy. The attacker can then identify the targeted person by physically observing who is present at the inferred locations over time.

Consequences: The privacy of the targeted person is violated if his diagnosis key is published. This could allow the attacker to harm, e.g. shame or extort, the targeted person. If this attack is done at scale, then the privacy of many diagnosed people would be violated. This could lead people to opt out of the system, diminishing its effectiveness.

Mitigation: The mobile device varies the signal strength[2] in a way that makes it hard to determine with good accuracy the location of the device from the few samples that the attacker may collect over a reasonably short period of time (e.g. a couple of minutes). This may require reducing the frequency of advertising. A determined attacker may be forced to stalk the targeted person, in order to collect more samples, and risk getting discovered.

5 Conclusions

We provided a security analysis of the COVID-19 Contact Tracing Specifications by Apple Inc. and Google Inc. The specified system is intended to help fight the COVID-19 pandemic while keeping user privacy and security central to its design. Our analysis focused on system security considerations yet also included information security considerations. We showed that the current specifications may introduce significant risks to society, such as loss of trust in the system, in COVID-19 diagnosis tests, or both. We analyzed power and storage drain attacks, relay and replay attacks, trolling attacks, and tracking and deanonymization attacks, as well as their consequences and mitigation strategies for them. The mitigation strategies we proposed are easy to adopt. In particular, they do not require major changes to the specifications, such as moving from unidirectional to bidirectional communication, as is the case in some of the previously proposed mitigation strategies that rely on a challenge-response protocol.

Acknowledgements

The author would like thank Mayank Varia for insightful comments that helped improve this report.

References

[1] Google Inc. Apple and Google partner on COVID-19 contact tracing technology. https://www.apple.com/newsroom/2020/04/apple-and-google-partner-on-covid-19-contact-tracing-technology/ retrieved April 12th, 2020

[2] Google Inc. Apple and Google partner on COVID-19 contact tracing technology. https://www.blog.google/inside-google/company-announcements/apple-and-google-partner-covid-19-contact-tracing-technology/ retrieved April 12th, 2020

[3] Apple Inc and Google Inc. Privacy-safe contact tracing using Bluetooth Low Energy. https://www.blog.google/documents/57/Overview_of_COVID-19_Contact_Tracing_Using_BLE.pdf retrieved April 12th, 2020

[4] Apple Inc and Google Inc. Contact Tracing Bluetooth Specification v1.1 https://www.blog.google/documents/58/Contact_Tracing_-_Bluetooth_Specification_v1.1_RYGZbKW.pdf retrieved April 12th, 2020

[5] Apple Inc and Google Inc. Contact Tracing Cryptography Specification https://www.blog.google/documents/56/Contact_Tracing_-_Cryptography_Specification.pdf retrieved April 12th, 2020

[6] Serge Vaudenay Analysis of DP3T Between Scylla and Charybdis https://eprint.iacr.org/2020/399.pdf

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

[8] Rivest et al. The PACT protocol specification https://pact.mit.edu/wp-content/uploads/2020/04/The-PACT-protocol-specification-ver-0.1.pdf retrieved April 12th, 2020

[9] Covid Watch http://www.covid-watch.org retrieved April 12th, 2020

[10] TCN Protocol https://github.com/TCNCoalition/TCN retrieved April 12th, 2020

[11] Anonymous Collocation Discovery: Harnessing Privacy to Tame the Coronavirus https://arxiv.org/pdf/2003.13670.pdf retrieved April 12th, 2020

[12] Decentralized Privacy-Preserving Proximity Tracing https://github.com/DP-3T/documents retrieved April 12th, 2020

[13] Chan et al. PACT: Privacy Sensitive Protocols And Mechanisms for Mobile Contact Tracing https://arxiv.org/pdf/2004.03544.pdf retrieved April 12th, 2020

[2] Recall that the specifications already define other Bluetooth metadata, such as the random address, as anonymized.

[14] Wikipedia - Mobile Phone Tracking https://en.wikipedia.org/wiki/Mobile_phone_tracking retrieved April 12th, 2020

[15] Wireless Positioning in IoT: A Look at Current and Future Trends https://tutcris.tut.fi/portal/files/16481836/sensors_18_02470_v2.pdf retrieved April 12th, 2020

[16] CaPSuLe: A Camera-based Positioning System Using Learning https://www.cs.rice.edu/~as143/Papers/CAPSULE.pdf retrieved April 12th, 2020

[17] Inferring User Routes and Locations using Zero-Permission Mobile Sensors http://www.ieee-security.org/TC/SP2016/papers/0824a397.pdf retrieved April 12th, 2020

A Self-Locating

In this section, we briefly discuss a number of methods for self-locating a mobile device relevant to contact tracing. A complete list of such methods is out of scope for this paper. Instead, our goal in this section is to demonstrate that there is a wide variety of existing techniques that can be employed to enable efficient location cell determination. We note that the operating system is in an excellent position to decide on the self-locating methods to employ. It can use its awareness of the hardware to optimize the selected combination subject to the available hardware. Using its complete access to the mobile device, it can make very informed trade-off decisions, such as between accuracy and power consumption. We leave this research direction to future work.

A number of factors make self-locating easier for the purpose of contact tracing:

1. It is sufficient to find a location cell, rather than accurate coordinates, that can be checked for in the matching procedure, as described in Section 3.2. In fact, it is also sufficient to find a small number of possible location cells that each can be checked; we will see below that this tolerance for location ambiguity can be useful for our purposes.
2. It is also sufficient to determine the location cell some time after sensory information is collected and associated (in private encryption form) with a proximity identifier received, rather than immediately after. This is because the matching procedure, which is the only function in the system that uses location cell determinations, is not invoked immediately upon receiving the proximity identifier. Thus, the location cell can be determined on-demand, and for many (non-matching) collected proximity identifiers such determination will never be needed; this saves power. Moreover, the mobile device will often have ample time for executing sophisticated location cell determination algorithms, even ones that utilize (secure) cloud services.
3. Because the specifications involve the operating system layer, the space of possible self-locating methods is generally larger than if they only involved the app layer. In particular, the operating system has unconditional direct access to all sensors on the mobile device whereas apps may have to obtain multiple user permissions to gain (possibly non-direct) access to required sensors. Of course, the operating system is trusted with this access, in particular for privacy purposes. Consequently, it is easy to employ a method that integrates multiple ones where each utilizes some of the available sensors.
4. Since only sensory information need be associated with a proximity identifier upon its reception, the operating system can perform this task automatically and with low power consumption, without waking up any app.

With these factors in mind, we proceed to discuss methods for self-locating.

A natural and well-known method for self-locating is using the Global Positioning System (GPS). While GPS self-locating is quite accurate, to about 5-15 meter radius (depending on reception conditions and quality of the GPS receiver), it has its disadvantages. GPS consumes a fair amount of power when turned on, and if turned off it takes a significant amount of time to reinitialize, enough for a person walking with the mobile device to change location by significantly more than the accuracy radius. Moreover, GPS is less useful for our purposes in urban areas because it is noticeably less accurate inside buildings and the 2-dimensional location it provides may not be enough for contact proximity purposes. Arguably, additional methods besides GPS should be considered.

Other well-known methods for self-locating include [14]:

- Network-based methods which utilize the service provider's network infrastructure.
- Handset-based methods which use client software.
- SIM-based methods which use raw radio measurements.
- Wi-Fi methods which use the characteristics of nearby Wi-Fi hotspots.

- Hybrid methods which utilize multiple approaches to locating by Wi-Fi, WiMAX, GSM, LTE, IP addresses, and network environment data.

More recently proposed methods for self-locating include:

- Power-efficient wireless positioning for Internet-of-Things (IoT) [15].
- Camera-based positioning [16] employing a locally-executed approximate image matching algorithm.
- Positioning using readings from the gyroscope, accelerometer, and magnetometer sensors[3] [17]. This method can result in some reasonable level of location ambiguity, which is not a problem for our purposes, as discussed above.

[3]While this method was presented as an attack, and access to the sensors has been tightened since, the operating system has unconditional access to the sensors and can employ the method as a desirable service, rather than as an attack.