Privacy and Security Risk Evaluation of Digital Proximity Tracing Systems

The DP-3T Project

19 April 2020

The basic idea behind digital proximity tracing through mobile applications is to use Bluetooth Low Energy (BLE) signals to estimate physical proximity between two smartphones. The only functionality that such an app needs to provide is to inform the contacts of an infected person that they might have been exposed to the virus through a close-range physical contact. The system **does not** need to reveal to anyone **who** the potential contagious contact was **with,** or **when** and **where** it happened.

This document summarises the findings of an in-depth privacy and security analysis of digital proximity tracing systems that our team has conducted over the past few weeks. It lists the risks inherent to any digital proximity tracing system, risks inherent to systems based on BLE handshakes between personal smartphones, and additional risks of proposed design variants of the latter. The risk analysis in this document was primarily conducted by the DP-3T team, but it is also informed by online discussions on the project's GitHub repository, ePrint reports, as well as exchanges via email and other platforms. We are grateful for all of this assistance and welcome suggestions about risks we missed.

Summary

In this document, we study privacy and security risks associated with deploying digital proximity tracing systems sometimes also referred to as contact tracing systems. Our main findings are:

(1) ***All proximity tracing systems*** that notify users that they are at risk enable a motivated adversary to identify the infected people that she has been in close proximity to. This risk is a consequence of the basic proximity tracing functionality and does not depend on any design choices or implementation details.
(2) ***In all practical proximity tracing systems based on Bluetooth measurements*** for proximity detection the following risks exist:
    (a) An adversary with a powerful antenna can trigger false alerts about encounters with an infected person that do not reflect real-world physical proximity.
    (b) An adversary can disrupt the contact discovery between users through noise injection in the radio channel.

   (c) An adversary can track users based on aspects orthogonal to contact tracing (e.g. MAC addresses).
   (d) *An adversary* that actively goes wardriving can identify locations with infected people present.

(3) ***All proximity tracing systems that communicate with a backend server*** might reveal the identities of infected people to the network provider and the server.

(4) ***In some decentralised proximity tracing systems that rely on infected individuals to share their identifiers*** for the purpose of contact tracing, the ephemeral identifiers of infected users can be used to trace these users for a limited period.

(5) ***In proximity tracing systems that* rely on infected individuals to share observed identifiers** for the purpose of contact tracing, the backend server might learn colocation information about infected users and their social interactions.

(6) ***In centralised proximity tracing systems*** in which a central server computes a user's risk score and notifies users at risk, the backend server, or any entity that gains access to the backend server, such as hackers or law enforcement agencies through subpoenas, is in a position to:
   (a) Track and target users based on their identifiers
   (b) Learn the interaction graph around infected individuals
   (c) Learn at-risk status of contacts of infected people

Table of contents

1. Privacy and security concerns

We consider the **privacy** of a system to be compromised if any entity is able to learn information about individuals or communities beyond the information needed to fulfill the basic functionality requirements of the system. For contact tracing, this includes revealing the identity of infected users, tracking a user's path, or learning about interactions between non-infected users. For a more detailed list of privacy concerns see the DP-3T whitepaper, Section 5.2.

A **secure** digital proximity tracing system should prevent malicious actors from jeopardising the confidentiality, integrity, and precision of the data reported to users. This means, for example, users should receive a notification of exposure if, and only if, a user was in close physical proximity to a contagious person. Other security concerns are attackers preventing users from learning that they have been exposed or recording contact events, see DP-3T whitepaper, Section 5.3, for more detailed explanations.

2. Taxonomy of risks

We consider three types of risks: inherent risks that apply to all contact tracing systems, generic risks that apply to systems using BLE, and risks that stem from the information exposed by the network.

	All PT systems	BLE-based PT systems	Systems sharing infected identifiers	Systems sharing observed identifiers	
			Decentralised	Decentralised	Centralised
	Section 2.1	Section 2.2	Section 3.2	Section 3.4	Section 3.5
Identify					
Infected individuals (IR 1)	✓ Multiple accounts	✓ Multiple accounts	✓ Eavesdropping	✓ Injection	✓ Multiple accounts
Locations with infected people present (GR 3)		✓ Multiple accounts	✓ Eavesdropping	✓ Injection	✓ Multiple accounts
Prevent notification (IR 2)	✓	✓	✓	✓	✓
Cause false alarms					
Through range extension (GR 1)		✓	✓ Injection	✓ Eavesdropping	✓ Eavesdropping
Through active relay (GR 2)		✓ Bi-directional	✓ Uni-directional	✓ Uni-directional	✓ Uni-directional
Disrupt contact discovery (GR 4)		✓	✓	✓	✓
Track a BT enabled device (GR 5)		(✓)	(✓)	(✓)	(✓)
Reveal app usage (GR 6)		✓	✓	✓	✓

FIGURE IR: **Overview of inherent and generic risks of digital proximity tracing systems.** For each type of system, the table lists whether the specific systems present this risk (✓) and if the risk can be mitigated (✓). The notes below the checkmarks specify, where applicable, the attack mechanism that leads to the risk.

2.1 Inherent risks of proximity tracing systems

In this section, we describe the privacy and security risks that are ***inherent to any proximity tracing system*** that fulfills the basic functionality of these systems**.** These risks are "inherent" because, to the best of our knowledge, they are present in all digital proximity tracing systems and do not depend on any specific implementation.

IR 1: Identify infected individuals

Any proximity tracing system that notifies users that they are at risk enables a motivated attacker to identify the infected people that he or she has been physically near. This risk is a consequence of the basic proximity tracing functionality. The attack can be executed regardless of implementation and proximity detection mechanism (BLE or otherwise). It only relies on the single bit of information that any proximity tracing system must reveal--whether you have been in close proximity to an infected person.

Attackers can learn the identity of infected people by combining two pieces of information: (1) *who* they interacted with at each time, and (2) that they were in close proximity to an infected person at a specific time. The first is information that the attacker must actively collect outside the app. The second is information that the attacker extracts from the app itself.

To know who she interacted with, the attacker keeps a log of personal interactions. To learn at which time she interacted with an infected person using the app, the attacker proceeds in two steps:

1. She creates multiple accounts in the proximity tracing system and uses them only for a short time (e.g., 15 minutes).

2. If a notification arrives, the attacker examines the corresponding account. Since this account was only used during a fixed time window, the attacker now knows that she was in close proximity to an infected person during that period.

In most systems that we have seen so far, creating multiple accounts is inexpensive, and the attacker can simply rotate accounts on the same device. In the worst case, the attacker might need to register each account on a different device. Using strong authentication (e.g., phone numbers or national identity cards) to limit account creation may raise privacy concerns for users.

With both sources of information, the attacker can assemble a list of individuals who are now potentially infected from the people in their log of personal interaction during the "contagious" time period. By combining information from multiple time windows, the attacker can narrow down their list to a small group of people and, in some cases, single out infected individuals.

The logical inferences necessary to conclude who is infected are easy to execute, and in fact, they might be possible without additional data gathering. Consider the following examples in which Eve tries to learn who in her environment is infected.

- Eve has been at home with her wife Alice for the past two weeks. Eve went out once to buy groceries from the grocery store around the corner. She was lucky: only the owner, Bob, was there. After a few days, Eve is notified that she was in close proximity to an infected person. She knows it is not Alice, and therefore concludes that Bob must be infected. *This inference did not require Eve to gather auxiliary information.*

- Suppose Eve has modified her system so that she learns, for every shop she visits, whether she was in close proximity to an infected person during that time. Eve is happy to go outside again and visits the butcher, the bakery, and the local bookstore. She is only notified about the butcher, and since she was the only customer in the store, she concludes that the employee working there must be infected.

- Suppose that Eve has two meetings today, and has modified her system so that she learns during which meeting she was in close proximity to an infected person. Suppose she meets with Alice, Bob, and Charlie in the morning and with Bob and Dave in the afternoon. Her system tells her she was in close proximity with an infected person in the afternoon. She concludes that Dave is infected (and not Bob, because then she would also have been notified about the morning meeting).

We emphasize that these attacks work against any contact tracing system, as they rely on the core proximity tracing functionality: notifying at-risk people. Without this notification, proximity tracing system would be useless. The risk is inherent to proximity tracing.

IR 2: Prevent notifications

In any contact tracing system, users, with benign or malicious intents, can ***ensure that (some) users are not notified that they are at risk even though they have been exposed***. To do so, the user either simply decides not to participate in contact tracing, temporarily disables the app (or Bluetooth functionality if that is what the app uses for contact measurements), or does not upload contact tracing data after being diagnosed.

This feature is needed to ensure that contact tracing apps are voluntary. Designing the system under the assumption that a part of the population will not participate enables the core principle of giving everyone the freedom to decide individually if they want to participate in tracing. The assumption that not everyone will participate, ensures that the app is effective in the presence of incomplete data (e.g., people without phones).

2.2 Risks of practical BLE-based systems

In this section, we describe privacy and security risks that are ***inherent to all practical proximity tracing systems based on BLE measurements*** for proximity detection. We call

these risks *inherent* because, to the best of our knowledge, all practical BLE-based systems incur these risks and specific implementations do not affect these risks.

The risks presented in this section are present in any radio-based proximity tracing system. While limiting these risks is an active research area (for example, the use of distance bounding protocols), these solutions are not yet ready to be rolled out. We therefore have to assume some risks to ensure deployability.

GR 1: Cause false alarms through BLE range extensions

All proximity tracing systems that rely on the exchange of BLE signals between personal smartphones as a proxy for physical proximity are susceptible to BLE range extension attacks that ***lead to users being falsely alerted that they are at risk***. This attack violates the integrity of the system since it reports a contact and potential infection that does not correspond to an actual close-range physical contact.

To cause false alarms in any proximity tracing system based on BLE, a malicious adversary simply places her proximity tracing device in a crowded area and hooks up a sensitive antenna and/or powerful transmitter to artificially increase the range of her Bluetooth contacts. As a result, other devices located beyond 2 meters can interact with the attacker's device and will perceive the attacker's device as "near-by". To complete the attack, the attacker must ensure that these interactions between her device and other devices are flagged as at-risk events. To do so, the attacker either:

1. **Is herself infected** and brings her device to the hospital when she gets tested (requiring the attacker to be infected).

2. **Pays a symptomatic person** to bring the attacker's device to the hospital instead of their own (or simply obtains the upload authorization code from them).

3. **Hijacks/bribes the health authority** that authorises infected individuals to trigger contact tracing.

4. **Hijacks/bribes the system server** that sends information or directly notifies users of the system. Most systems we consider here use a backend server to check authorizations by health authorities and to relay information to users. As a result, the attacker can also collude with or bribe the backend server to help generate fake contact events.

Some of these attack vectors can be mitigated to a certain degree:

- Designs that bind upload authorisation to the actual data being shared, for example through commitments to specific dates or identifiers *at the time of testing*, makes approach 2 more costly for the attacker, because the attacker must "swap phones" (and pay) before testing, but does not have the guarantee that the patient will test positive. If authorisation codes are not specific to upload data,

adversaries can simply pay people who have already received a positive diagnosis to transfer their authorisation code to the adversary. The adversaries then upload their own malicious data. This mitigation is not possible in designs that upload observed ephemeral identifiers as these are not yet all known at the time of testing and can hence not be committed to.

- To reduce trust (and therefore the risk of bribing/collusion/hacks) in the backend server, one could either let the health authority (who must always be trusted to not to cause fake contact events) operate the server, or let the health authority sign the data the server distributes.

To summarise, it is impossible to avoid Bluetooth range extension attacks on proximity tracing systems that rely on Bluetooth handshakes as a proxy for physical contact. This is due to the fact that one cannot distinguish a-posteriori between BLE signals broadcast by a device carried around by an actual person and a signal emitted by a powerful transmitter or captured by a long-range antenna.

GR 2: Cause false alarms through active relays

Proximity tracing systems that rely on exchange of BLE signals between personal smartphones as a proxy for physical proximity are very likely to be susceptible to BLE active/real-time relay attacks that ***lead to user's falsely being alerted that they are at risk***. This attack violates the integrity of the system since it reports a contact and potential infection that does not correspond to an actual close-range physical contact.

To cause false notifications**,** the attacker would actively relay the Bluetooth signals of people that the attacker believes will soon be diagnosed with SARS-CoV-2. For example, the attacker could observe and **relay** Bluetooth signals from people at a testing center.

Such an attack could in theory be mitigated by including a distance bounding protocol to prevent such active relay attacks or by binding transmissions to locations (requires location services and location permission, and possibly active exchanges between phones). We do not expect these counter measures to be deployable within the next few months.

GR 1 and GR 2 address the same harm, but describe different attacks that can lead to this harm. In GR 1, the attacker uses her own device normally but extends its range. Therefore, the attacker needs to find a way to mark her device as infected. In GR2, the attacker simply relays interactions with devices that are likely to be marked as infected without intervention from the attacker.

GR 3: Identify location with infected people present

A risk similar to risk IR 1 is an attacker who tries to ***identify locations with infected people present***. Such an attack can be particularly effective at identifying the homes of infected people, especially when performed at night, when people are likely to be at home.

To do so, the attacker actively goes *wardriving* and uses his device (or multiple devices) to instead map *locations with infected people*. The attacker would use a directional antenna, possibly in combination with a powerful antenna (compare with the GR 2 range extension attack above) to target individual houses or locations. The attacker then uses the techniques from the inherent identification attack (IR 1) to identify which of these locations house infected people.

The attack's effectiveness can be reduced by requiring the attacker to be in proximity for a longer time. Systems using passive broadcasts could, for example, use secret-sharing of identifiers so that the attacker must listen or broadcast for a few minutes to be able to conduct the attack. Systems using active connections could instead require a minimum connection duration.

GR 4: Disrupt contact discovery

In ***all proximity tracing systems based on BLE-based distance measurements***, an attacker with a Bluetooth jammer can disrupt communication between normal users of the system, and therefore ensure that ***close proximity events cannot be established*** by the respective users.

Rather than prevent contact events altogether, an attacker can also try to artificially create a large amount of contact events, potentially flooding the memory of a user's phone, or requiring the phone to not store some of the contact events it detects.

GR 5: Tracking a Bluetooth enabled device

The designs we consider use BLE for proximity detection. Using the tracing application and therefore enabling Bluetooth brings some fundamental risks:

- Enabling Bluetooth can make the device trackable if the OS does not implement MAC address randomization and disables advertisements.

- Bad synchronization between MAC address randomization and Bluetooth identifiers makes a device trackable as long as the attacker stays within range.

The first point is addressed in modern smartphone operating systems. The second point would be solved by the Apple/Google proposal.

GR 6: Reveal usage of the contact tracing app

The designs we consider use BLE for proximity detection. Using the tracing application, enabling Bluetooth, and transmitting tracing-app specific identifiers will reveal to any observer that the user has the tracing app installed. This information is not considered particularly sensitive as participation in digital proximity tracing will be seen as contributing to the social good in many societies.

2.3 Risks of networked systems

Finally, we discuss a class of ***risks that originate from the use of a communication network.*** These attacks can typically be mitigated, but we wish to point them out here, because they are likely to apply to a large class of systems.

NR 1: Network identities reveal data about infected patients

Any proximity tracing system in which infected individuals upload data directly from their phone to a central server, ***reveals to a system administrator or the central server which individuals have tested positive*** through their associated network identifiers. This attack is generic in the sense that all systems that use direct upload functionality are vulnerable to it.

The (network) identities of infected patients can be hidden by a simple proxy that relays the uploads from phones to the central server. For example, each local hospital could serve as a proxy for data uploads to avoid traffic analysis attacks. This approach suffices as we *must trust the hospital with the privacy of infected patients.*

NR 2: Traffic analysis reveals data about infected patients

Any proximity tracing system in which infected individuals upload data directly from their phone to a central server without counter measures, ***reveals to a on-path network observer that a patient uploaded data to the central server***.

Most proposals for proximity tracing systems assume that soon after a user of the application receives a positive test result, she will upload the information necessary to trigger contact tracing from her personal device to a central server. This enables an on-path network eavesdropper, for example a curious internet service provider, WiFi provider, or backbone operator, to learn of the user's infection. It also enables the central server to obtain a pseudo-identity for the infected person.

A proxy does not help to mitigate this attack. However, if the data is small, users could regularly upload dummy packages, e.g., empty messages of the same size as a real report, to the server. The server will simply ignore these dummy packages. Since users use an encrypted connection to the server, network observers cannot distinguish these dummy packages from real uploads, thus hiding their infection status even from network observers.

3. Analysis of specific design points

In this section, we explore different design points of BLE-based contact tracing systems. We explore these different design points in an hierarchical manner, see Figure OO. For each class of systems, we reexamine the risks from previous sections if necessary, and explore additional risks that these design points entail.

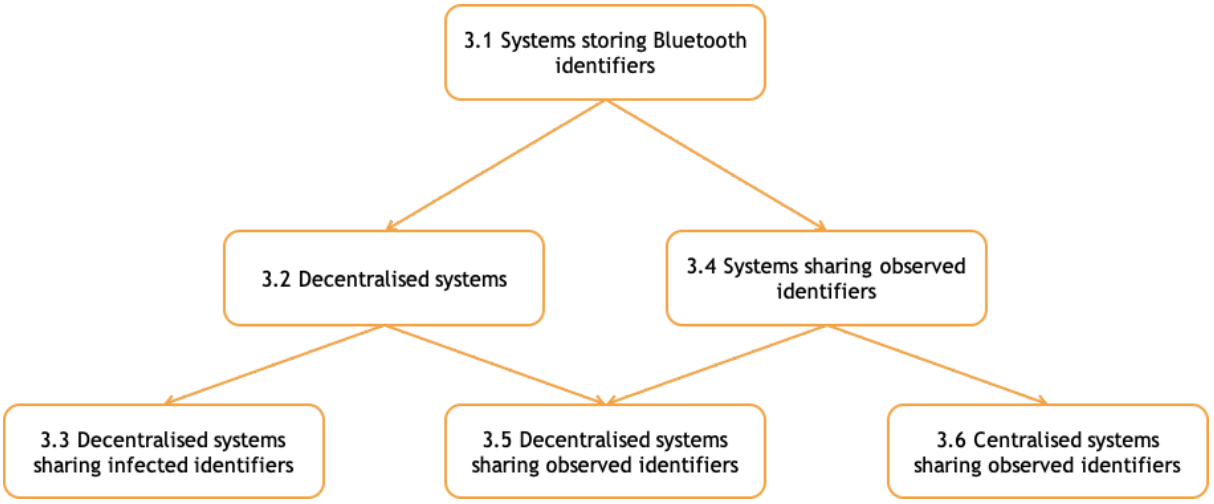

Figure OO: **Overview of design points discussed in the following sections.**

	Systems storing BT observations	Systems sharing infected identifiers	Systems sharing observed identifiers	
		Decentralised	Decentralised	Centralised
	Section 3.1	Section 3.3	Section 3.5	Section 3.6
Reveal social interactions				
Through local phone access (SR 1)	✓	✓	✓	✓
To a central server (SR 5)			✓ Infected users	✓ Infected users
Recompute risk score (SR 2)	✓	✓	✓	✓
Location tracing				
Through local phone access (SR 3)		✓	✓	
By other users (SR 4)		✓/✗ Infected users		
Through access to a central server (SR 7)				✓
Reveal colocation to a central server (SR 6)			✓ Infected users	✓ Any user (SR 8)
Reveal social graph (SR 8)				✓
Reveal at-risk status (SR 9)				✓

Figure SR: **Overview of system-specific risks for different designs of digital proximity tracing systems.** For each risk, we indicate whether the risk is present (✓) or not (✗) and, where applicable, indicate if the risk applies to all users or only to specific subgroups.

3.1 Risks of systems that store observed Bluetooth identifiers

We refer here to a large class of ***designs in which phones locally store observed Bluetooth identifiers***. Centralized systems, such as the ROBERT candidate for PEPP-PT, and the decentralized systems proposed by the DP-3T consortium or Apple/Google, are examples of such systems.

System-specific risks

SR 1/SR 2: Reveal social interactions and recompute risk score through local phone access

A law-enforcement adversary (LEA) or local attacker (e.g., an abusive spouse) can obtain access to a victim's phone, either legally by a subpoena or through direct coercion. All designs considered in this section locally store observed Bluetooth identifiers. If an attacker gains access to such information, this poses a two-fold risk.[1]

- (SR 1) ***Reveal social interactions***
    - The adversary learns how many Bluetooth identifiers were collected in a given time-frame. This serves as a proxy for the number of people somebody was with.
    - The adversary can use this information to confirm that the device has been in close proximity to a third party given the Bluetooth identifier of that party.
- (SR2) ***Recompute the risk score given the recorded observation, possibly using different parameters***
    - The adversary can use the information stored locally on a phone to recompute the risk score of the phone's owner without the explicit consent of the individual.
    - This could, for instance, lead to discrimination against individuals.

This **local** attack applies equally to phones of infected and non-infected users. To minimize the effectiveness of this attack, systems should minimise the data stored on a device to the minimum amount necessary.

We emphasize that this attack requires access to the device *and* the technical knowledge to extract data from it.

[1] Serge Vaudenay, "Analysis of DP3T", 2020, Cryptology ePrint Archive, Report 2020/399

3.2 Risks of decentralized systems

We refer to designs in which ***each device has control over their own key generation and the contact tracing process happens locally as decentralised.*** Examples of such systems are the proposals by the DP-3T consortium and Apple/Google.

Revisiting inherent and generic risks

An inherent risk ***(IR 1)*** in any proximity tracing system is that a motivated attacker learns who is infected. In decentralised systems, there is usually an easier way for an attacker to learn *when she was in close proximity to any infected person* without creating multiple accounts. In all decentralised designs we have evaluated, the information shared with a device to enable local risk computation is sufficient for an attacker to determine ***when*** she was in contact with an infected person and use this information to reveal the identity of the infected. In Sections 3.3 and 3.5 we give detailed examples of attacks that lead to this risk.

System-specific risks

SR 3: Location tracing after local phone access

Law-enforcement agencies or local attackers (e.g., an abusive spouse) can obtain access to a victim's phone, either legally by a subpoena or through direct coercion. If an attacker gains access to the information stored on the device pertaining observed Bluetooth identifiers and broadcasted identifiers, the attacker can learn broadcasted Bluetooth identifiers of the victim, enabling ***location tracing of the victim*** given other observations, or confirm the location of the victim in the past.

This attack applies equally to phones of infected as of non-infected users. To minimize the effectiveness of this attack, systems should store the minimum amount of data, and only for as long as is necessary. The use of a "master key" from which a device's daily keys are derived (as in the v1.0 Google/Apple design) would allow law-enforcement to link an individual's identifiers for the **entire lifetime of the application**.

We again emphasize that this attack requires access to the device *and* the technical knowledge to extract data from it.

3.3 Risk of decentralised systems that share infected identifiers

In this section, we list risks in decentralised proximity tracing systems ***in which an infected user shares (a compact representation of) their own broadcast identifiers*** over the contagious period with other devices for the purpose of contact tracing. The proposals by the DP-3T consortium or Apple/Google are examples of such systems.

Linkability of infected identifiers

In designs that require infected individuals to share their broadcast identifiers used during the contagious window, these broadcast identifiers might become ***linkable***, i.e., other entities learn ***which broadcast identifiers belong to the same infected person***.

If phones of infected users send data directly to the server, the server has the ability to link identifiers for the entire duration of the contagious period if multiple Bluetooth identifiers are associated to the same upload/network identifier. This can only be prevented if uploads go via a proxy, for instance via a local hospital server. In this case, the server can only link identifiers to the extent that other users can (see below).

The extent to which other users can link identifiers of infected individuals depends on how identifiers of infected people are represented when they are shared. We have seen three approaches, which differ in how many Bluetooth identifiers can be linked to a single infected individual.

- **One seed for the entire contagious period.** In these designs (e.g., the DP-3T low-cost design) infected people share a single seed that enables others to reconstruct, and thus link, an infected person's identifiers for the entire contagious period.

- **One seed per medium-length time window.** In these designs (e.g., the joint proposal by Apple and Google), infected people share independent per-day (or other comparable time windows) seeds so that identifiers are linkable per day but not across days during the contagious period.

- **Independent Bluetooth identifiers.** In these designs, all Bluetooth identifiers are independent (e.g., as the unlinkable DP-3T design), to prevent any linkability of ephemeral identifiers of infected users.

The ability to link identifiers belonging to the same infected person can be used by adversaries to facilitate new attacks on decentralised designs or strengthen attacks. In particular, as we show below, being able to link identifiers of infected people makes them easier to identify and enables tracking.

Revisiting inherent and generic risks

An inherent risk ***(IR 1)*** in any proximity tracing system is that a motivated attacker learns who is infected. In decentralized systems in which infected people share their identifier, there is an easier way for an attacker to learn when she was in close proximity to an infected person without creating multiple accounts. The attacker can simply match the set of infected identifiers against each of her recorded Bluetooth identifiers to determine ***when*** she was in contact with an infected person and use this information to reveal the identity of the infected.

When a design allows an attacker to ***link*** identifiers of the same person (see above), the attacker can ***combine observations at different times to identify the infected person***. For example, the attacker might learn that the infected person he saw at 10:11AM is *the same* as the one he saw at 14:14PM. While he might have encountered many different people at each time, the intersection might be much smaller. This further increases the likelihood that the attacker can successfully single out an infected individual.

The original identification attack and the variant above use fine-grained timestamps associated with observed Bluetooth identifiers. ***A retroactive attacker who did not modify her phone*** to collect this information will only have access to coarse-grained time information for each Bluetooth observation (e.g., because generated identifiers are shuffled as in the low-cost DP-3T design, not visible outside of the OS as in the Apple/Google proposal, or simply not recorded as in the unlinkable DP-3T design). This coarse granularity makes identifying infected individuals more difficult. However, retroactive attackers can *still use linked identifiers* to combine multiple coarse-grained observations to further narrow the list of potentially infected people.

A generic risk of any BLE-based contact tracing systems ***(GR 1)*** *is* an attacker that uses BLE range extension to make others falsely believe that they were in close proximity to an infected person. In the decentralized design we consider here, the attacker must ***broadcast*** her identifiers at high power so that they will be recorded by many other phones (even those far away). The attacker must have broadcasted for a period long enough for other devices to conclude that the attacker was in proximity enough time to constitute a risk.

The risk ***(GR 2)*** of false at-risk notifications through active relays assumes an attacker that relays messages to/from a (soon to be) infected person. In the systems considered here, the attacker must ***observe ephemeral identifiers of people that might be infected and broadcast these elsewhere***. Depending on the specific design, the attack works even if there is some delay between observing Bluetooth identifiers and **replaying** them elsewhere. For the DP-3T low cost design, this delay is limited by the coarse time window (hours or days), whereas for the Apple/Google proposal and the DP-3T unlinkable design, the replay must happen within an epoch (e.g., within 15 minutes).

System-specific risks

Revisiting SR 3: Location tracing through local phone access

The risk of location tracing described in Section 3.2 above stems from the fact that phones locally ***store their own broadcast identifiers*** alongside the *observed* broadcast identifiers. The former enables location tracing of a victim if a third party gains access to the phone. However, in decentralized systems that share infected identifiers, the user's own identifiers need not be stored on the phone in the clear**.** Instead, they can be stored in encrypted form, so that only the backend server can decrypt them after the patient has been diagnosed with SARS-CoV-2. This partially mitigates the risk SR 3.

SR 4: Location tracing of infected individuals

Adversaries that are able to link broadcast identifiers that belong to the same infected individual can leverage this information ***to track a user's path over the contagious period***. The tracing is limited to the contagious window for which infected share their identifiers and for the duration for which identifiers become linkable in the respective design (see above). In particular, that means that ***infected people in the unlinkable design are not traceable.***

3.4 Risks of systems that share observed identifiers

In this section, we analyse designs in which each device locally stores the set of observed identifiers and then ***shares this set of observed identifiers to enable contact tracing***. The attacks discussed in this section ***equally apply to centralised and decentralised systems that follow this approach***. In later sections, we further split these systems into decentralized designs (see Section 3.5) and centralized designs (see Section 3.6). The ROBERT and PEPP-PT systems are examples of centralized designs that locally store observed identifiers and then upload these identifiers to the server for central risk scoring.

Revisiting inherent and generic risks

A generic risk of any BLE-based contact tracing systems ***(GR 1)*** *is* an attacker that uses BLE range extension to make others falsely believe they were in close proximity to an infected person. In the group of designs we consider here, the attacker must receive Bluetooth identifiers from phones that are further than 2 meters away. To do so, the attack can use sensitive antennas to receive Bluetooth identifiers sent by others.

The risk ***(GR 2)*** of false at-risk notifications through active relays assumes an attacker that relays messages to/from a (soon to be) infected person. In the systems considered here, an attacker must observe identifiers of target people whom she wants to notify and broadcast their identifiers to individuals likely to be tested positive. This way the attacker ensures that the target people's identifiers will be included as contacts in the list of observations the infected person will upload. Depending on the specific design, the attack works even if there is some delay between observing identifiers and replaying them elsewhere.

System-specific risks

SR 5: Reveal social interactions to a central server

In systems in which infected individuals share the set of identifiers observed during the contagious period, the ***central server can learn information about the social interactions of infected individuals***. The central server learns

- How many Bluetooth identifiers an infected individual collected during the contagious period. This data serves as a proxy for the number of people somebody

came close to.

- That the patient was in proximity to a third party given the Bluetooth identifier of that party.

The information obtained by the central server is similar to the information obtained by a local adversary that accesses a user's phone (see SR 1).

Depending on how the patient communicates the observed identifiers to the server (i.e., via a proxy or not; using a permanent identifier or not), the server can associate this information to a specific person.

SR 6: Reveal colocation information about infected individuals to a central server

In the designs considered here, an infected person uploads all identifiers observed during the contagious window to the server. For epochs in which groups of at least three people were in close proximity to each other, this will ***reveal temporal colocation information about infected individuals to the server.***

For instance, suppose Alice, Bob, and Charlie were in close proximity on Monday morning. If Alice and Bob are diagnosed with Sars-CoV-2, then they will both upload Charlie's identifier for Monday morning. Hence the server will conclude that Alice and Bob (or their long-term pseudonyms) were co-located.

3.5 Risks of decentralized systems sharing observed identifiers

We refer to decentralized designs in which each device has control over their own key generation and the contact tracing process happens locally. In this section, we list attacks on decentralised proximity tracing systems in which ***an infected user shares, for the purpose of contact tracing, the list of broadcast identifiers observed during the contagious period*** with other devices.

A note on efficiency. We wish to point out that the set of observed identifiers is likely to be much larger than the set of own broadcast identifiers. And that the download cost of mobile devices may therefore become an issue, in particular, as the interaction rate between people goes up.

Revisiting inherent and generic risks

An inherent risk ***(IR 1)*** in any proximity tracing system is that a motivated attacker learns who is infected. In decentralized systems in which infected people share their observations, there is an easier way for an attacker to learn when she was in close proximity to any infected person without creating multiple accounts. All the attacker needs to do in the class of designs we consider here is to ***frequently rotate her own Bluetooth broadcast identifiers*** and keep a log of the people she saw with fine-grained timing information. Later, when the attacker receives a list of identifiers observed by an

infected person, she can see which of her Bluetooth identifiers were observed by infected people, and therefore in which time period she interacted with an infected person.

We note that the **frequency of key rotation** is under the control of the attacker. Even though normal users only rotate their broadcast keys every, say, 15 minutes, the attacker could choose to use a new key every minute to get more fine-grained information.

As long as the design is able to store past broadcast identifiers with coarse timing information only, then, as for the other decentralized approach, it is more difficult for a retroactive attacker to gather fine-grained timing information.

3.6 Risks of centralized designs that share observed identifiers

We now consider BLE-based designs in which the backend server determines who is at risk and thus needs to be able to map observed identifiers to permanent identifiers. The simplest way to build such a design is to let infected people upload the list of observed broadcast identifiers and duration. The server then looks up the owner of each of these observed broadcast identifiers and notifies them if their risk score is above a threshold. In other variants of centralised systems, such as ROBERT, devices query the backend server to learn about their risk status.

The key property here is that to identify at-risk people ***the server must be able to associate ephemeral Bluetooth identifiers to long-term identifiers that can be used to notify the corresponding device owner***.

The DP-3T Project has published a more specific analysis of a centralized protocol, PEPP-PT-NTK, elsewhere.[2]

Pseudonymity of users

The central server can associate long-term identifiers with ephemeral Bluetooth identifiers. Therefore, from the point of view of the server, users are pseudonymous. It is well known that pseudonyms can easily be related back to actual identities.

Risk of data breaches and data leaks

Central data collection and processing, as envisaged by the proposed centralised proximity tracing systems, introduce the inherent risk of data breaches and data leaks. A poorly secured system that stores sensitive information about a large number of users presents a target for motivated adversaries.

[2] The DP-3T Project, '*Security and privacy analysis of the document 'PEPP-PT: Data Protection and Information Security Architecture*' (19 April 2020)

Therefore, even if the designers of such a system are willing to accept the risk of the server learning privacy-sensitive data, the analysis should also account for the risk of unauthorized access to this data by third parties.

Revisiting inherent and generic risks

An inherent risk ***(IR 1)*** in any proximity tracing system is that a motivated attacker learns who is infected. This risk is inherent. The only method to address the risk of an attacker that frequently rotates between accounts is to limit the number of different accounts/devices a user can use. However, preventing sybils — fake accounts controlled by the same user — while at the same time allowing legitimate uses is very difficult. Small payments, captchas, proofs of work, etc. are all easily circumvented by a motivated attacker. It is possible to instead rely on government-issued identity documents to prevent the creation of sybils. However, this very possibly enables the server to associate real world identities to account registrations, which exacerbates the other attacks we describe below.

Therefore, we conclude that the inherent identification attack remains possible in this centralized design.

System-specific risks

SR 7: Location tracing through access to a central server

Law-enforcement (with a subpoena) and operators of the backend server can associate a long-term identity to ephemeral Bluetooth identifiers. This would enable them to ***trace the location of any user of the system given Bluetooth observations***. In particular, it allows them to perform a location validation attack, as with the decentralized designs above, without accessing any phone.

Furthermore, it also allows an adversary that colludes with the server to trace the ***future movements of any individuals*** for which the attacker has observed at least one Bluetooth identifier or knows the permanent identifier of the target. Moreover, if the server assigns identifiers to users of the system, the server could even assign constant, or easily recognized identifiers to make tracing users possible for law enforcement, or other actors, ***without access to the backend database.***

These two mechanism have far-reaching consequences for users:

- **Deanonymization users given a single Bluetooth identifier**. The adversary can easily de-anonymize a target by either linking a set of observed identifiers to the same person or by providing them with distinguishable Bluetooth identifiers that make them recognizable.

- **Long term persistent surveillance of individuals**. To enable **long-term surveillance of selected users by third parties,** the backend could either selectively reveal all future keys of these users, or instead assign special Broadcast keys to them. (Note

that a user has no way to know whether their key changes over time or not.) We note that this also works for communities, as one could assign specific identifiers to a target group of people.

SR 8: Reconstructing social interaction graphs

The centralized system ***reveals the interaction graph around each infected user to the backend server***. This is by design, as the server maps each time-stamped ephemeral identifier back to a permanent pseudonym to enable contact tracing.

The subset of the full interaction graph learned by a server grows quickly as every newly infected user uploads their entire contact history, which can be linked to existing nodes in the graph. Exposure of this information is an even greater problem due to the characteristics of COVID19. Contacts of an infected individual are expected to become infected in a short period of time and the system must perform near-real-time, fine-grained tracking of physical interactions. Therefore, the backend learns not only isolated graphs of sick people, but more importantly, connected components that *quickly provide an accurate representation of social structures and users' interactions for a much wider population*.

Moreover, as a result of this, **even people who have not been infected will have (a large portion of) their social graphs exposed by others' submission of contacts that include them**.

See Figure AA for an example. As the infection spreads, the server learns more and more social connections of user O_2 even though that user never uploaded any data. The nodes in the graph are pseudonymous, however it is well known that rich graph data can easily be reidentified. In particular, in the setting considered here where the server is likely to have access to additional information that helps to re-identify users.

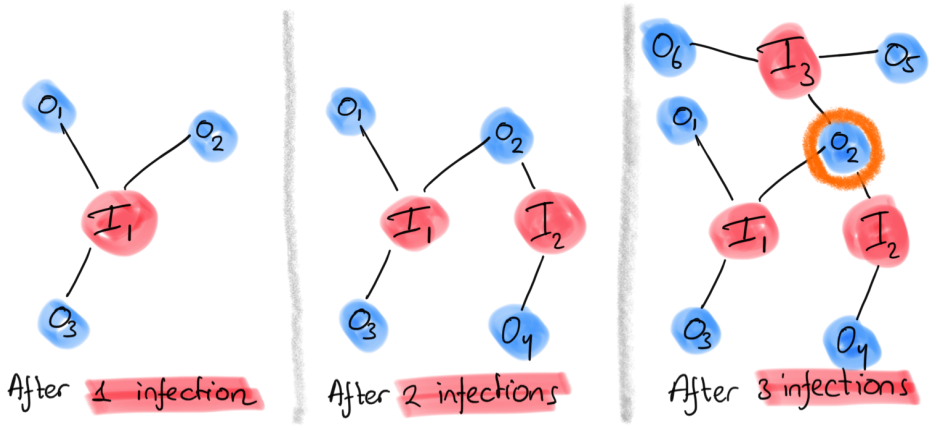

Figure AA: After 3 observations, the server knows a lot about at-risk person O_2.

We also note that rich graphs, such as the one that is created in the backend, are prone to de-anonymization attacks with little auxiliary information.[3] A recent study noted, for example, that knowing 1% of the network around individuals would mean observing 46% of all communications and quickly re-identifying a large fraction of the population.[4] This leads to re-identification of individuals, non-infected and infected.

In comparison to SR 6, where the server only learns colocation information of infected people, in a centralised system the server ***in addition learns colocation information about non-infected people .***

SR 9: Reveal at-risk status to a central server

By construction, the backend notifies users that are at risk because they have been in close physical proximity to at least one infected person. As a result, the backend ***learns who (or at least which pseudonyms) are now at risk***.

[3] See for example: Ji et al. "Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization." 24th USENIX Security Symposium (USENIX Security 15). Or Narayanan, Arvind, and Vitaly Shmatikov. "De-anonymizing social networks." *2009 30th IEEE symposium on security and privacy*. IEEE, 2009. Or Sharad, Kumar, and George Danezis. "An automated social graph de-anonymization technique." *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. 2014.

[4] Radaelli et al. "Quantifying surveillance in the networked age: Node-based intrusions and group privacy." arXiv preprint arXiv:1803.09007 (2018).