Hansel and Gretel and the Virus*

Privacy Conscious Contact Tracing

Jaap-Henk Hoepman[1,2]

[1] Radboud University Nijmegen, Email: jhh@cs.ru.nl
[2] University of Groningen

Abstract. Digital contact tracing has been proposed to support the health authorities in fighting the current Covid-19 pandemic. In this paper we propose two centralised protocols for digital contact tracing that, contrary to the common hypothesis that this is an inherent risk, do not allow (retroactive) tracking of the location of a device over time. The first protocol does not rely on synchronised clocks. The second protocol does not require a handshake between two devices, at the expense of relying on real-time communication with a central server.

We stress that digital contact tracing is a form of technological solutionism that should be used with care, especially given the inherent mass surveillance nature of such systems.

1 Introduction

The road to hell is paved with good intentions. This is perhaps especially the case in exceptional times like these, as the world is suffering from a global pandemic caused by the Severe Acute Respiratory Syndrome CoronaVirus 2 (SARS-CoV-2). The pandemic has spurred the development of *contract tracing* apps, that aim to support the health authorities in their quest to quickly determine who has been in close and sustained contact with a person infected by this virus [36, 22].

Contact tracing (also known as *proximity tracing* or *exposure notification*) apps have been in use in China for some time [27]. The main idea underlying digital contact tracing is that many people carry a smartphone most of the time, and that this smart phone could potentially be used to more or less automatically collect information about people someone has been in close contact with. The work of Feretti *et al.* [11], modelling the infectiousness

* Version: Sun Feb 21 22:37:48 2021 +0100 / arXiv2-4-gf558ca9 / hansel-and-gretel.tex

of SARS-CoV-2, showed that digital contact tracing could in principle help reduce the spread of the virus, under a number of strong assumptions:

- a significant fraction of the population use the app,
- the app has a very low false negative rate,
- a large fraction of infected people are successfully isolated, and
- the delay between establishing a contact, determining whether that person is indeed infected, and getting that person to (self)-isolate is short (1 day or less).

The often cited threshold of 60% of people that need to install the app for digital contact tracing to be successful [16] seems to be based on the rather optimistic assumption that people go into isolation without delay (0 days), there is a false negative probability less than 0.1, and that 70% of infected people successfully isolate (looking at the data from Feretti *et al.*). This is not to say that some reduction in the spread of the virus can already be observed at lowed adoption rates [18].

Notwithstanding these rather optimistic assumptions, and with testing capacity initially insufficient to even test people working in high risk environments (like health care professionals), Bluetooth-based contact tracing apps have quickly been embraced by governments across the globe as a necessary tool to ease or end the lock-down as imposed in many countries in 2020. Such Bluetooth based contact tracing apps broadcast an ephemeral identifier on the short range Bluetooth radio network at regular intervals, while at the same time collecting such identifiers transmitted by other smartphones in the vicinity. The signal strength is used as an estimate for the distance between the two smartphones, and when this distance is determined to be short (within 1–2 meters) for a certain period of time (typically 10–20 minutes) the smartphones register each others ephemeral identifier as a potential risky contact. Some, like the European Commissioner for Internal Market Thierry Breton, even went so far as to call them 'deconfinement' apps[3].

This is a dangerous form of 'technological solutionism' [26]: instead of thinking the underlying problem through, we go for a quick technological fix to counter the symptoms. This is by no means a Luddite argument to forego the use of technology per se, but rather a warning that "what typically separates good from bad practice is adherence to rigorous, contextual testing and working within existing expertise, institutions and approaches to augment what we know works, rather than to work around or challenge

[3] In a tweet on April 22, 2020, https://twitter.com/ThierryBreton/status/1253039000782286848.

the integrity of those experts or knowledge" [23]. Technological support for contact tracing works much better when it is actually developed in close coordination with the health authorities [28].

The model of Feretti *et al.* is based on a very incomplete picture of how the virus spreads exactly. Even worse, the product lead of the TraceTogether app used in Signapore, warns that no Bluetooth contact tracing system deployed or under development, anywhere in the world, is ready to *replace* manual contact tracing [1]. The problem being that to establish whether two people that were in close contact could indeed have infected one another depends on context, like whether the place they met was properly ventilated. Such information about a context in which people met cannot be derived or even approximated using Bluetooth (or other commonly available sensors). A human-in-the-loop is therefore necessary.

Privacy by design is another good intention that paves the road to this particular hell. Even though many have proposed privacy conscious designs for a contact tracing system (see section 6), such systems *are* a mass surveillance tool by definition. Which makes the framing of such apps as deconfinement apps all the more worrisome, even cynical. Especially in cases where the system's primary purpose is to take care of our health and well-being, there is a tendency to lower the bar and accept greater level of (state) surveillance as a form of 'pastoral power' [13, 15]. As noted by Taylor *et al.* [29] "the pandemic has amplified a nascent epidemiological turn in digital surveillance". Attempts to use the pandemic as justification for extending surveillance powers should be resisted [10].

Notwithstanding these reservations, this paper will discuss yet another privacy conscious design for a contact tracing app. (And because of these reservations I hesitate to write privacy friendly in this context.) Why study contact tracing at all, given these reservations, one might well ask. There are several answers to that question. First of all, and what got us started on the subject many months ago, was the purely engineering question of whether contact tracing could be done with some level of privacy consideration in the first place. Second of all, it was clear from the start that governments and health authorities would consider digital technologies to support their work in controlling the spread of the virus. Resisting the ill-advised, perhaps desperate, reach for a technological quick fix is only a first line of defence. If that defence fails, we had better finished a comprehensive map of the design space and have the least problematic solution ready for them to grab.

1.1 Contributions

A common distinction among contact tracing apps is whether *all* contacts are registered *centrally* (on the server of the national health authority for example) or in a *decentralised* fashion (on the smartphones of the users that installed the contact tracing app) [22].[4] In the first case, the authorities have a complete and perhaps even real time view of the social graph of all participants. In the second case, information about one's contacts is only released (with consent) when someone tests positive for the virus.

Our first contribution is to show that there actually exists a third semi-centralised category, between centralised and decentralised that *only* provide the health authorities with the identities of those people that have been in close contact with an infected person.[5] We present two solutions to the contact tracing problem that belong to this category. The first protocol is peer-to-peer based, but requires an actual exchange of messages between two devices before a contact is registered. On the other hand this protocol does not allow other phones (or the authorities for that matter) to later retroactively track the movements of a device over time. This refutes the hypothesis (specifically, risk SR7 identified in [30]) that this is an inherent, systemic, risk of all centralised contact tracing systems. The second protocol does not require a handshake between two devices, at the expense of relying on real-time communication with a central server and requiring synchronised clocks to prevent relay attacks.

In the course of our construction we also discuss a new cryptographic primitive called *replay resistant encryption* that prevents replay of encrypted messages in anonymous settings where senders cannot prove freshness of messages with a signature over a nonce.

We study semi-centralised instead of decentralised solutions to the contact tracing problem for two reasons. First of all, the discussion of the strong assumptions under which contact tracing could help reduce the spread of the virus we offered above indicates that a centralised solution (where the health authorities obtain a full and immediate picture of all people that have been in contact with an infected person) is preferable over a decentralised

[4] Note that essentially all systems for contact tracing require a central server to coordinate some of the tasks. The distinction between centralised and decentralised systems is therefore *not* made based on whether such a central server exist, but based on where the matching of contacts takes place.

[5] The DESIRE protocol, discussed in section 6 is similarly semi-centralised, and was developed in parallel (when the first draft of this paper was written in March 2020, see https://blog.xot.nl/2020/03/25/hansel-and-gretel-and-the-virus-privacy-conscious-contact-tracing/index.html).

approach (that relies on the initiative of people to report to the health authorities themselves once the system has notified them of having a risk of infection). (Semi-)centralised solutions also offer the option to speed up the contact tracing process [35]. Second of all, the perceived privacy benefit of decentralised solutions is debated [33].

The paper is structured as follows. We first describe the main source for inspiration of this research, namely Apple's 'find-my-iPhone' feature, in section 2. We then formalise the contact tracing problem in section 3. After that we present and analyse two protocols in section 4 and section 5. Section 6 discusses related work. We finish with our conclusions in section 7.

2 Source of inspiration

Our solutions described below are inspired by solutions to a related problem, namely that of locating lost devices. In particular, we use ideas present in Apple's 'find-my-iPhone' feature that allows lost devices without a GPS sensor (like certain iPads or iWatches) to be located as well. This system supposedly[6] works as follows[7].

The system distinguishes four different entities. L is the lost device. P is the trusted partner device the lost device was previously paired with. This is typically a different device owned by L's owner. People are assumed to immediately pair new devices with devices they already own: without this earlier pairing, lost devices cannot be found. H represents any of the millions of helper devices out there that are used by the system to locate lost devices. Finally, C is the central cloud storage used by the helpers to store the location of (lost) devices they came in contact with, and that is used by the trusted partner devices to retrieve the location where their lost devices were last seen.

Device L has a secret identifier id_L known to P. Paired device P has a public key K_P known to L. L uses its secret identifier id_L to generate a random pseudonym rid_L, for example by hashing the identifier and the current time t using a cryptographic hash function h, i.e., $rid_L = h(id_L, t)$. L broadcasts this random pseudonym every once in a while over its local Bluetooth and/or WiFi interface, in the hope that some helper H (e.g. anybody else's iPhone that happens to be nearby) receives this signal. Randomising the identifier is necessary to prevent anybody from singling out and tracking L

[6] No official documentation about how this feature is implemented appears to exist.

[7] See https://blog.cryptographyengineering.com/2019/06/05/how-does-apple-privately-find-your-offline-devices/ (Accessed 2020-03-25).

through its static identifier id_L. (This assumes the MAC address of the Bluetooth or WiFi interface is properly randomised as well, which is not always the case [21].)

This helper H is supposed to know its current location (because it does have a GPS receiver) and is supposed to send its location and the pseudonym of the device it discovered to Apple's iCloud servers. The trusted partner P trying to locate L will query these servers with the recently used pseudonyms for L using its knowledge of id_L that allows him to compute $rid_L = h(id_L, t)$ for any t in the recent past. If a tuple for any of these queries exists, the locations contained in it will be reported back as a result, allowing P to locate L.

In order to protect the location privacy of the helper H, the location it reports to Apple's iCloud needs to be encrypted (otherwise Apple would know the location of all helpers), and it needs to be encrypted against P's public key so P can actually decrypt it. The only way for H to know this key, is if the lost device L broadcasts it alongside its randomised pseudonym. But as P's public key is static, this would serve as an ideal static identifier to track L, thwarting the use of the randomised pseudonym. This can only work if L can also randomise P's public key K_P before broadcasting it. Luckily, encryption schemes exist that allow the public key to be randomised while ensuring that the same unchanged private key is the only key that can be used to decrypt a message [34].

It is worth noting that this protocol appears, at first sight, to be vulnerable to devices that *pretend* to be lost but instead are planted at specific locations. Such planted devices could lure helpers to encrypt their location to a 'fake' public key K_P broadcast by them, which is in fact controlled by the authorities that have the corresponding private key k_p. Luckily, Apple never releases the identity of the helper to the paired device. In other words, the scheme works because it is entirely under Apple's control. We need to trust Apple anyway as Apple could, at any time, decide to harvest the location of all devices it manufactures by a simple operating system update.

3 Problem definition

Contact tracing is one of the traditional tools used to fight an epidemic. The goal of contact tracing is to find all recent contacts of a patient that tested positive for infection by a virus. The idea being that these contacts could have been infected by this patient. By tracing these contacts, testing them, and treating or quarantining them, spread of the virus can be contained [36]. Contact tracing is typically done 'by hand' by the national

health authorities, and is a time consuming process. Digital contact tracing is supposed to support this traditional form of contact tracing, as people may not necessarily know or remember all people they have recently been in contact with.

Informally speaking, a system for contact tracing should help the health authorities to find all people that have been in close contact to a person carrying an infectious disease (like COVID-19), with the understanding that having been in close contact for a certain period of time with such a patient creates a significant enough risk of being infected as well. A contact is considered 'close' if two people were less that several meters away from each other (denoted d further on) for a certain duration (denoted T further on). Such a contact is only relevant if it occurred at most Δ days before a person got tested positive for infection (which includes a possible delay between being infected and actually getting tested positive).

As many (but certainly not all) people carry a (smart) mobile phone with them almost all the time, it is natural to consider the use of such devices to *automatically* collect data for contact tracing[8]. As the precision required to establish closeness is in the range of at most several meters, GPS or mobile phone location data cannot reliably be used for this. Currently considered solutions for automatic contact tracing therefore quickly converged to the use of Bluetooth, even though even that technology has serious limitations [7]. For non-automatic contact tracing, QR-code based solutions are being proposed[9] that require users to explicitly scan QR codes of buildings they enter, or on the phone of other people they meet.

3.1 Assumptions

We assume a system of *mobile devices*, that move around freely, and that are always active. In the following we will assume smartphones as the primary devices.

Each device X has a unique *identifier* id_X, only known to itself, that it obtains by enrolling in the system. This identifier is inextricably linked to

[8] A low tech, non automatic, approach would be to ask people to keep track of all other people they meet in a small (paper) diary, as initially suggested by Prime Minister Jacinda Ardern of New Zealand, see: https://www.theguardian.com/world/2020/apr/19/jacinda-ardern-asks-new-zealanders-to-keep-diaries-to-help-trace-coronavirus-contacts.

[9] E.g. the NZ COVID Tracer app, see: https://www.health.govt.nz/our-work/diseases-and-conditions/covid-19-novel-coronavirus/covid-19-novel-coronavirus-resources-and-tools/nz-covid-tracer-app, or Zerobase, see: zerobase.io

the unique owner of the device (that may get infected, or may need to be tested for infection). We therefore identify an owner with the identifier of their[10] device. In particular we write 'infected device' X for a device X that belongs to an infected person. We assume the authorities can contact the owner when given this identifier. Identifiers are secret and cannot be forged or guessed.

Each device is equipped with a *short range broadcast (radio) network* with which it can detect all other devices in the vicinity and communicate with them. In the following we will assume a Bluetooth network. It is assumed that all devices within maximal 'infection distance' d can be reliably detected and reached, and that the distance can be reliably measured over this network. That is to say, a message m broadcast over the radio network reaches all devices within d of the broadcasting device. Devices receiving this message as the tuple $\langle m, d\rangle$, where m is the message received, and d is the distance between the recipient and the sender[11]. We do not assume that receivers are able to identify the sender of a message, or that they are able to determine whether two messages received were sent by the same device (unless the contents of the messages themselves would provide that information). This models the case where Bluetooth networks use frequently changing random MAC addresses. Devices can accurately measure passage of time; we do not assume synchronised clocks for the first protocol.

We furthermore assume a *central server* (operated by the health authorities A) with a public key K_A and corresponding private key k_A. All devices know the public key K_A of the authorities. Devices can reliably communicate in real time with this server using a *long range data network*. In particular, the server is authenticated (but the devices are not for privacy reasons), and the communication is encrypted. In the following we assume an Internet connection based on a cellular data network or a WiFi connection.

3.2 Cryptographic primitives, and replay resistance

The protocols below rely on two different semantically secure public key encryption schemes, for example based on elliptic curve cryptography.

One of these schemes, denoted $\widetilde{E}_K(m)$, is used to encrypted messages broadcast on the Bluetooth channel. For this a 'raw' ElGamal based system based on Curve25519 [2] is an appropriate choice, offering the equivalent

[10] We use them/their to avoid the clumsy "he or she" or "his or her" gender neutral expressions.

[11] We wish to stress, again, that this commonly made assumption is not at all realistic in practice.

of 128 bits 'symmetric' security [9, 20]. With this choice, keys are 256 bits (32 bytes) long, and so is the message space. Encryption is done using ElGamal directly (and not in a hybrid fashion where ElGamal is used to encrypt a random symmetric key that is then used to encrypt the message). Note that this means messages first need to be encoded as a point on the curve, see e.g., [14]. Ciphertexts are still semantically secure under the Decisional Diffie-Hellman assumption [32], but note however that this traditional method of using ElGamal directly is malleable and not IND-CCA (i.e., indistinguishable against chosen ciphertext attacks [19]). This is not a problem in our protocols. The other scheme, denoted $E_K(m)$, is an arbitrary strong IND-CCA cipher.

A cryptographic hash function $h : H \mapsto H$ is also used. SHA-3 could be used for this [12].

One particular concern in our protocols is to prevent message replay attacks that try to create false positives (see the "authenticity" requirement in section 3.5). A standard way to ensure freshness of messages is to let the sender sign a fresh nonce (either provided by the receiver, or derived from the current time). It is not clear how to do so in the setting studied here were devices are supposed to remain anonymous (unless they have been in contact with an infected device).

In the first protocol below, devices R broadcast their public key K_R and expect to receive information from nearby devices encrypted against this key. Adversarial devices may broadcast their own key, receive information from nearby devices encrypted against this key, decrypt this information and rebroadcast it (now encrypted against the key of another unsuspecting device) in a relay attack attempt. The specific information sent by a device in our protocols is actually its identity encrypted against the public key of the authorities, e.g., $E_{K_A}(id_B)$. We make use of the involvement of the authorities to implement *replay resistant encryption*, a construct that may be of independent interest.

In replay resilient encryption we distinguish three entities: an anonymous sender S, a relay R (with key pair k_R, K_R) and the authorities A (with key pair k_A, K_A). The goal is to let A accept a message m sent by S, but only if it was indeed received by R directly from S. We write $\Xi_{R;A}(m)(m')$ (where m' is a message that is received by R in the process; R does not learn m).

One way to implement this primitive is to define

$$\Xi_{R;A}(m)(m') = \widetilde{E}_{K_R}(E_{K_A}(m \,\|\, h(K_R)) \,\|\, m') \; .$$

Relay R decrypts this message to obtain m' and $M = E_{K_A}(m \,\|\, h(K_R))$. It forwards M to A for decryption, together with its public key K_R. A then uses his

private key k_A and the public key K_R just received to verify the decryption of M, rejecting it if this does not contain $h(K_R)$.

A malicious adversary Z trying the attack outlined above would receive $\widetilde{E}_{K_Z}(E_{K_A}(m \,\|\, h(K_Z)) \,\|\, m')$, which it can decrypt to obtain $E_{K_A}(m \,\|\, h(K_Z))$ and m'. Trying to replay this message against another device R means sending $\widetilde{E}_{K_R}(E_{K_A}(m \,\|\, h(K_Z)) \,\|\, m')$, which R can indeed decrypt, but which would fail the later test performed by A (as $h(K_Z) \neq h(K_R)$). This prevents Z from replaying m and getting it accepted by A.

If relay R regularly changes keys and discards old private keys, then a replay attack where an adversary first collects keys broadcast by R and then later re-broadcasts these keys in the vicinity of the victim will not work either. By the time the adversary has returned back to the relay R, it already discarded the private key necessary to decrypt the replayed messages.

3.3 Definition and functional requirements

For two devices B and C define $B \approx_t C$ to be true when B and C have been within distance d of each other for at least $T + 2\delta$ time at time t.[12] Define the predicate $\mathsf{contact}(B, C)$ to be true when $B \approx_t C$ for some t within the last Δ days. By definition $B \approx_t C$ implies $C \approx_t B$, and hence $\mathsf{contact}(B, C)$ implies $\mathsf{contact}(C, B)$.

Let device B maintains a set $\mathsf{contacts}_B = \{C \| \mathsf{contact}(B, C)\}$, the set of all devices that B was in contact with (within the last Δ days).

We can distinguish two different schemes for contact tracing: a *centralised* one, and a *decentralised* one (see figure 1). A centralised scheme for contact tracing allows the health authorities A to obtain $\mathsf{contacts}_B$, but only with the active cooperation of a device B itself, and presumably only when the owner of device B tested positive for the virus. A decentralised contact tracing system on the other hand allows device B to notify all devices C such that $C \in \mathsf{contacts}_B$, but only when B was instructed by the health authorities A to do so.

A decentralised contact tracing system notifies all contacts in $\mathsf{contacts}_B$ of an infected person owning device B. Such a decentralised contact tracing system is more aptly called a *exposure notification* system instead. In a decentralised contact tracing system the health authorities do not learn who has been in contact with an infected person. However, people notified of such an exposure can be instructed to contact the health authorities, so that

[12] The additional time 2δ is the leeway we have to allow for the protocols to detect a contact, as it is infeasible in practice to monitor for presence on the Bluetooth channel continuously: this would quickly drain the battery.

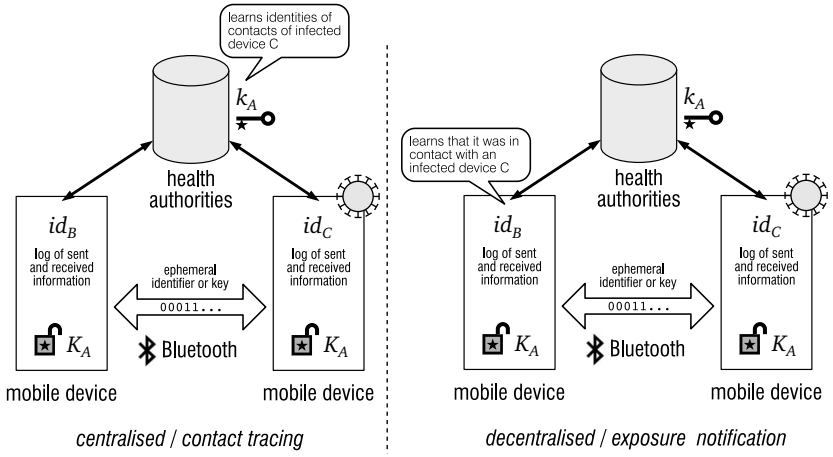

Fig. 1. System model: centralised contact tracing versus decentralised exposure notification.

the health authorities would obtain the same information regardless. In the remainder of this paper we only consider centralised schemes for contact tracing.

We have the following functional requirements for such a contact tracing system [31] (where we write 'infected device' for the device of a user that tested positive for infection ans subsequently consents to let their device share contact tracing information).

completeness If $\mathsf{contact}(X, Y)$ then $Y \in \mathsf{contacts}_X$. Whenever device X becomes infected, the health authorities A learn id_X and all id_Y such that $Y \in \mathsf{contacts}_X$ (preferably without post-contact cooperation of Y).

precision/soundness The health authorities A only learn id_Y when Y itself becomes infected or when $\mathsf{contact}(X, Y)$ for an infected device X (within the last Δ days measured from the moment user X tested positive and consented to sharing information).

In other words, when $Y \notin \mathsf{contacts}_X$, the health authorities do not learn id_Y. Note that precision of the contact tracing system (i.e., the fact that membership of $\mathsf{contacts}_X$ corresponds to actual physical proximity for an epidemiologically relevant time period) is pretty much assumed in the remainder of this paper (but please bear in mind the discussion on this important topic in section 3).

3.4 Threat model

Apart from the devices, their users, and the health authorities we have to reckon with the following additional entities.

Firstly, there may be other devices with access to the (Bluetooth) radio network that are able to eavesdrop on messages exchanged between devices, or that can insert messages of their own. We call these entities *adversaries*. We assume that adversaries do not have a valid identity, nor can they fake one from a genuine device.[13] Adversaries are *malicious*.

The contact tracing functionality is implemented by an app running on the devices. We assume the app design and implementation is fully open source and the binary is created using a reproducible build system from these sources. This means we can trust the app to be honest (in so far as we trust the protocols it implements, and that we believe the source is thoroughly scrutinised), but its users could be curious (trying to extract information collected by the app in order to track particular users, or to find out who is infected). In other words devices are *honest but curious*.

Note that actual *manufacturers* are responsible for building the devices and providing them with an operating system. These manufacturers are responsible for allowing the contact tracing app to be installed and for the app to access the short range (Bluetooth) radio network (and possible other sensors, data sources and network devices). Although in practice the long range data network (i.e., the Internet connection) actually reveals long term network identifiers (like IP addresses) that can be used to track or (re)identify devices, we assume this is not the case here.[14]

3.5 Security and privacy requirements

Given this threat model, we have the following privacy and security requirements [31, 30, 6, 5] for a centralised contact tracing system.

[13] Genuine devices may of course also assume this role, but for simplicity we then assume that these are two independent devices.

[14] There currently is no way to deal such identifiers in practice if one wants to preserve privacy. A VPN is too weak (the VPN provider sees everything its users do), yet Tor[8] is too strong (there is no need to protect against a NSA like adversary) given the impact on performance. This is yet another example that shows we direly need an efficient, frictionless, way to provide sender anonymity on the Internet, similar to the use of randomised MAC addresses on local networks.

confidentiality of infection Only the health authorities A learn which devices are infected. In particular, adversaries cannot learn whether (other) devices are infected or not.[15]

confidentiality of infected contact Suppose X is infected. Apart from the health authorities A no entity learns about a device Y such that $Y \in \mathsf{contacts}_X$.

confidentiality of uninfected contacts Suppose X is not infected. No entity learns about a device Y such that $Y \in \mathsf{contacts}_X$ (provided Y is not infected). In particular, X does not learn id_Y for any of its 'contacts' Y.

location privacy Adversaries cannot track a device while it moves around from one physical location to another,[16] either in real-time, or retroactively after additional information is released because of a device becoming infected. This corresponds to systemic risk SR7 identified in [30]. In this paper we show this risk *does not* apply to our protocols (and hence is not really systemic).

authenticity An set of adversaries $\mathscr{M}$ cannot convince a device X that $Y \in \mathsf{contacts}_X$ for some device Y for which there is no t such that $X \approx_t Y$, unless there are $M, M' \in \mathscr{M}$ such that both $X \approx_t M$ and $M' \approx_t Y$ for some time t. In other words, adversaries can only create false positives by relaying information.

3.6 Other considerations

There are other (ethical) concerns and considerations that need to be addressed[17] when implementing a system for digital contact tracing [25]. These are related to accessibility to the technology used for contact tracing (not everybody has a suitable modern smartphone), whether use of such technology is mandatory, the consequences of having been in close contact of a person that tested positive, or simply the risk of using Bluetooth signal strength as a rather poor proxy for being in close contact [7]. There will be

[15] Note that this stronger than claiming that adversaries cannot learn the identity id_Y of infected devices: it also requires them to not be able to single out such a device. This is considered an inherent risk of all exposure notification systems as shown in [30], but not of contact tracing systems.

[16] For example when trying to use the information devices broadcast over the short range broadcast (Bluetooth) radio network in order to detect close by devices and record such contacts for later. It is known that when phones broadcast fixed identifiers, if as few as 1% of the phones in London would have software to intercept these identifiers, this would allow an attacker to track the location (in a resolution of one square kilometre) of about 54% of the city's population [24].

[17] See e.g., https://rega.kuleuven.be/if/tracing-tools-for-pandemics

many false positives. These will create 'boy-cries-wolf' effects: people being flagged as having been in contact with the virus, but not developing any symptoms, may ignore warnings in the future. Stories about large numbers of such wrongfully flagged people will drive down compliance. Strong enforcement to counter this may backfire.

Some more general requirements for or constraints on contact tracing platforms have been voiced over the past few months. First and foremost is the requirement that the authorities should demonstrate that there is a clear need for the interference with fundamental rights caused by a system like contact tracing, and that this interference is proportionate. This could be done by basing the design on explicit recommendations from the health authority on how to effectively curb the spread of infectious diseases, embedding a system of contact tracing in a larger framework for epidemic control that includes effective testing procedures and quarantine measures. Many more steps can be taken to strengthen the proportionality of the system proposed. For example, a data protection impact assessment should be performed. The system could be developing in consultation with data protection authorities and other stakeholders (including representatives from civil society). Moreover, the designs and implementations should be open. The terms of service should be easily understandable, and there should be a continuous and public mediation/explanation of what is processed, and how. There should also be clear controls for people to exercise their data protection rights, e.g., the right to delete data once the epidemic is over, or to see who your data has been shared with.

4 A peer to peer protocol

We are now ready to present the two centralised contact tracing protocols. The first protocol (discussed in this section) exchanges messages between devices only to create records of proximity. It has one major drawbacks: it involves a handshake between both devices (which creates the risk that a failed handshake causes the proximity of both devices to not be registered). This is resolved by the second protocol, discussed in section 5.

Each phone C maintains a local log $\mathscr{L}_C[d]$ containing information about other phones detected in the vicinity d days ago, for $d \in \{0, \ldots, \Delta\}$, where Δ is the maximum number of days for which keeping this data is relevant.

At the start of each day, $\mathscr{L}_C[\Delta]$ is discarded, $\mathscr{L}_C[d] := \mathscr{L}_C[d-1]$ for $d = \Delta$ down to 1, and $\mathscr{L}_C[0] := \varnothing$. In other words the log rotates and is pruned daily to remove old entries that are no longer relevant.

The full protocol for phones B and C meeting each other now runs as follows (see also figure 2). The same protocol runs simultaneously with the

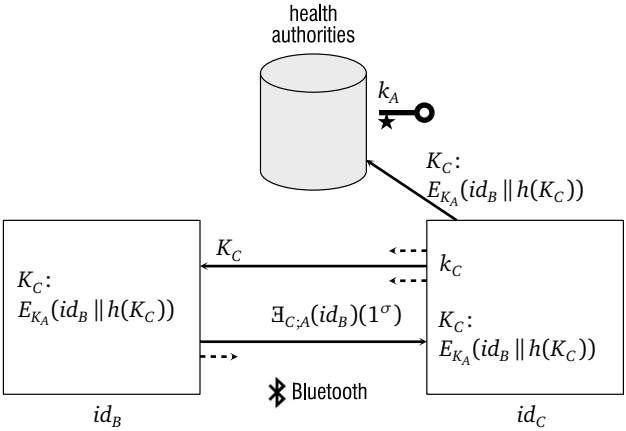

Fig. 2. The peer-to-peer protocol.

roles of B and C reversed (and for any other phone that also happens to be in the vicinity). Let σ be a sufficiently large constant, e.g., $\sigma = 16$. And let δ be some fraction of T. The protocol broadcasts Bluetooth 'beacons' every δ seconds.

- Whenever its MAC address changes,[18] C creates a fresh private key k_C and corresponding public key K_C. The previous public key is destroyed immediately; the previous private key after $T + \delta$ seconds (see below).
- C broadcasts this public key over the Bluetooth network as $\langle \mathbf{key}, K_C \rangle$ every δ seconds.
- When B receives a $\langle \mathbf{key}, K_C \rangle$ message, it estimates (based on signal strength) whether the message was sent by another device within unsafe distance d. If this is the case it stores it in a local pool of keys, together with the time t it first received it. As soon as it received this public key, B encrypts its identity id_B against the public key K_A (of the authorities) and stores this encrypted identity $E_{K_A}(id_B \,\|\, h(K_C))$ along with K_C in the pool. Keys in this pool received more than $T + \delta$ seconds ago are discarded, together with their associated data.
- Every δ seconds, for every public key K_C currently still in this pool, B does the following.
  - B retrieves the previously stored encryption of its identity for this public key and adds some redundancy 1^σ that will allow C to detect

[18] Note that this is not necessarily easy to do in practice.

successful decryption of a message (see below), and encrypts the message using K_C.
  - B broadcasts $E = \exists_{C;A}(id_B)(1^\sigma) = \widetilde{E}_{K_C}(E_{K_A}(id_B \,\|\, h(K_C)) \,\|\, 1^\sigma)$ over the Bluetooth network as $\langle \textbf{iam}, E \rangle$.

– For every $\langle \textbf{iam}, E \rangle$ messages it receives C, C first estimates (based on signal strength) whether the message was sent by another device within unsafe distance d. If that is the case it tries to decrypt it using its current set of private keys k_C (the ones that it generated at most $T + \delta$ ago), discarding any results that do not end with 1^σ.
– If successful, it stores the result $E_{K_A}(id_B \,\|\, h(K_C))$ in its log $\mathscr{L}_C[0]$ for today (together with a copy of K_C), but only if it received at least another copy of the same message at least T seconds ago.[19] Note that the log is a set, so duplicates are removed.

When someone is diagnosed as being infected, their identity id_C as well as the log $\mathscr{L}_C[\,]$ is extracted from their phone C. The system can be designed such that this requires physical presence of the phone at the site testing for infection, making sure the data is released only after an appropriately authenticated request. This request should come from the medical authorities A or from some independent oversight board. The log $\mathscr{L}_C[\,]$ is sent to the authorities, who can decrypt the entries $E_{K_A}(id_B \,\|\, h(K_C))$ using the private key k_A to obtain the set of all identities id_{B_i} that have been in the vicinity of C in the last Δ days. A verifies that the hash of K_C is indeed present to thwart replay attacks.

B relies on C to discard old log entries, and to only record its encrypted identity if it was at an unsafe distance for more than T seconds.

After releasing its log, the app locks itself: it no longer records any data. Reinstalling the app to reactivate it is (made) impossible. We are assuming here that once a person has been shown to be infected, he or she will stay immune and therefore not spread the virus further. Thus tracing who has been close to this person is no longer necessary.

4.1 Analysis

Define $\mathsf{contacts}_X$ as the set of all Y such that $E_{K_A}(id_Y \,\|\, h(K_X))$ in $\mathscr{L}_X[\,]$. We analyse the requirements outlined in section 3 one by one.

[19] Observe how we here use the fact that B encrypts its identity id_B against the public key K_A (of the authorities) *once* for every K_C it receives, and stores this encrypted identity $E_{K_A}(id_B \,\|\, h(K_C))$ along with K_C in the pool.

Completeness We have to show that if $\mathsf{contact}(X, Y)$ then $Y \in \mathsf{contacts}_X$, and that whenever device X becomes infected, the health authorities A learn id_X and all id_Y such that $Y \in \mathsf{contacts}_X$.

The first part is shown as follows. Suppose $\mathsf{contact}(X, Y)$. This means that X and Y have been within distance d of each other for at least $T + 2\delta$ time at some time t. X broadcasts K_X every δ seconds, so Y has received K_X at time $t-T-\delta$ at the latest. This means that during the time interval $[t-T-\delta, t]$ the pool of Y contains K_X and X has kept a copy of the corresponding private key k_X. Every δ seconds within this time interval, Y broadcasts $E = \Xi_{X;A}(id_Y)(1^\sigma) = \widetilde{E}_{K_X}(E_{K_A}(id_Y \,\|\, h(K_X)) \,\|\, 1^\sigma)$ which X is assumed to receive as it is within distance d during this interval. X can decrypt this message as it kept a copy of the corresponding private key k_X. The first and last such message X receives during this interval are guaranteed to be at least T time apart, and are guaranteed to be equal as Y encrypts its identity id_Y against the public key K_A (of the authorities) *once* for every K_X it receives. This guarantees that X includes $E_{K_A}(id_Y \,\|\, h(K_C))$ in $\mathscr{L}_X[\,]$ as required.

The second part follows from the fact that once X is diagnosed as infected, its log $\mathscr{L}_X[\,]$ is uploaded to the authorities. As this contains all entries $E_{K_A}(id_Y \,\|\, h(K_X))$ such that $Y \in \mathsf{contacts}_X$, and the authorities can decrypt this using their knowledge of k_A, the result follows.

Precision/soundness We have to show that the health authorities A only learn id_Y when Y itself becomes infected or when $\mathsf{contact}(X, Y)$ for an infected device X (within the last Δ days measured from the moment user X tested positive and consented to sharing information).

First observe that according to the protocol, only infected devices X send their identifier (and their logs) to the authorities directly. Moreover, a device Y only sends its identifier to other devices whose key K_X was recently received (these keys are discarded after $T + \delta$ seconds) while within the unsafe distance d. The identity id_Y is sent encrypted, first against the key of the authorities, and then again using this key K_X. Any device that has not been within this unsafe distance of Y will therefore not be able to decrypt this message.

Note however that a device X will already add $E_{K_A}(id_Y)$ in its log $\mathscr{L}_X[0]$ for today if it receives two copies of the same message at least T seconds apart. This is a strong indication that X and Y were within unsafe distance d during that whole period, but does not guarantee that during that period X and Y were actually that close all the time. Precision therefore only holds approximately.

Confidentiality of infection We have to show that only the health authorities A learn which devices are infected. This immediately follows from the fact only when someone is diagnosed as being infected, their identity id_X as well as the log $\mathscr{L}_X[\,]$ is extracted from their phone X and sent to the authorities, and that no information (about infection status) leaves the authorities.

Observe that the authorities may learn how often device Y was in close contact with device X during the day (or, at the very least, on how many different days).

Confidentiality of infected contact Suppose X is infected. We have to show that, apart from the health authorities A, no entity learns about a device Y such that $Y \in \mathsf{contacts}_X$.

We defined $\mathsf{contacts}_X$ as the set of all Y such that $E_{K_A}(id_Y \,\|\, h(K_X))$ in $\mathscr{L}_X[\,]$. As observed above, $\mathscr{L}_X[\,]$ is only shared to the authorities and clearly X itself knows $\mathscr{L}_X[\,]$.

If we can show that even X learns nothing about a device Y such that $Y \in \mathsf{contacts}_X$, then no other entity can either.

Note that all X receives are $\Xi_{X;A}(id_Y)(1^\sigma) = \widetilde{E}_{K_X}(E_{K_A}(id_Y \,\|\, h(K_X)) \,\|\, 1^\sigma)$ messages which it can decrypt only if the correct key K_X is used. This yields $E_{K_A}(id_Y \,\|\, h(K_X))$.

Confidentiality of uninfected contacts Suppose X is not infected. We have to show that no entity learns about a device Y such that $Y \in \mathsf{contacts}_X$. This follows from the analysis in the previous paragraph.

Location privacy We have to show that an adversary cannot track a device while it moves around from one physical location to another.

A device X either broadcasts a random key K_X, or an encrypted identifier $\widetilde{E}_{K_Y}(E_{K_A}(id_X \,\|\, h(K_Y)) \,\|\, 1^\sigma)$.

Regarding the key, the protocol guarantees/assumes it is refreshed whenever the MAC address of the Bluetooth channel changes. In other words, broadcasting this key does not create an additional vulnerability on top of the location tracking issues associated with MAC address randomisation [21]

Regarding the encrypted identifier, note that for any entity not equal to Y (to whose key the identifier is encrypted) the message is essentially random as the ElGamal encryption used is semantically secure (see section **??**). Device Y can decrypt (and any device in X vicinity can send its own random key and receive and decrypt) this message to obtain $E_{K_A}(id_X)$. Depending on the cipher used for this encryption (semantically secure or not), Y can observe a certain device to be in its vicinity during the day or not (see the discussion earlier on confidentiality of infected contact).

Authenticity We have to show that a set of adversaries $\mathscr{M}$ cannot convince a device X that $Y \in \mathsf{contacts}_X$ for some device Y for which there is no t such that $X \approx_t Y$, unless there are $M, M' \in \mathscr{M}$ such that both $X \approx_t M$ and $M' \approx_t Y$ for some time t.

This trivially follows from the fact that for any device Y, the value id_Y is secret (it is only revealed to the authorities) and cannot be guessed. The authorities never reveal any information they receive. Devices only broadcast their identity using replay resistant encryption. This ensures that no other entity ever obtains another device identity in plaintext, and that the authorities do not accept a replayed encryption. Therefore old messages that are replayed will be ignored, and ensures that we are in a relay scenario where $M, M' \in \mathscr{M}$ such that both $X \approx_t M$ and $M' \approx_t Y$ for some time t.

5 Involving the help of a central server

The peer-to-peer protocol presented in the previous section has a significant drawback: the protocol involves a handshake where phone C broadcasts a random public key and phone B responds with some encrypted information. If either of these messages fail to arrive, a possible contact remains undetected. In other words, the protocol is error-prone.

The following protocol solves these issues by involving the help of a central server. This help comes at a price though: more care has to be taken to ensure that the authorities do not learn more than necessary.

Let $h : H \mapsto H$ be a cryptographic hash function from a domain H to a range H.

As before each phone C maintains a local log $\mathscr{L}_C[d]$ containing information about the keys it used d days ago, for $d \in \{0, \ldots, \Delta\}$. This log rotates and is pruned daily to remove old entries that are no longer relevant, as described for the previous protocol.

To reliably detect close proximity for longer than the safe period T, time is divided into epochs ϵ_i that start at time $i \cdot T$ and last for T seconds.[20] If two device are found to have been in close proximity in two consecutive epochs, this is assumed to be a strong indicator that there is a significant risk of infection. This less exact method of establishing the duration of a particular contact is used to prevent the central server of the authorities from determining when exactly this contact took place.

[20] Some form of synchronisation of clocks between devices is needed to prevent replay attacks. The closer the synchronisation, the harder replay becomes. See the discussion later on.

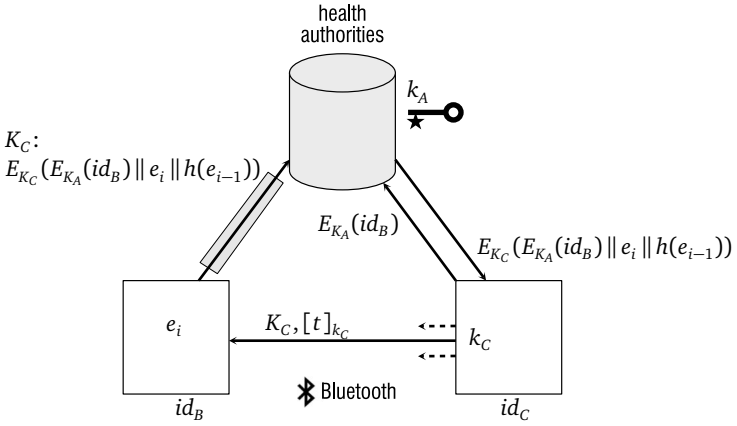

Fig. 3. The protocol involving the central server.

The full protocol for phones B and C meeting each other now runs as follows (see also figure 3).

- B generates a random value $e_i \in_R H$ for each epoch ϵ_i; it keeps this random for one more epoch and then discards it.
- Whenever its MAC address changes, C creates a fresh private key k_C and corresponding public key K_C. It stores these keys in the local log $\mathscr{L}_C[0]$ for today.
- C regularly (at least every δ seconds) broadcasts $\langle \mathbf{key}, K_C \rangle$ over the Bluetooth network. If replay must be detected, it also sends the current time, signed with the current private key $[t]_{k_C}$. The drawback is that in this case the broadcast no longer fits in a single Bluetooth beacon.
- B, when receiving such a broadcast, first estimates (based on signal strength) whether the message was sent by another device within unsafe distance.
- If this is the case, and the message contains a signature $[t]_{k_C}$, B verifies this signature with the public key K_C just received, and if the verification is successful, checks that the time t is equal to its own estimate of the current time.
- If this is the case, B encrypts its identity id_B using the public key K_A (of the authorities), it adds the current epoch random e_i and the hash $h(e_{i-1})$ of the previous epoch random first and then encrypts the result using the public key K_C it just received.
- B sends the result $E_{K_C}(E_{K_A}(id_B) \,\|\, e_i \,\|\, h(e_{i-1}))$ together with the public key K_C to the authorities A using the cellular network (again encrypted

against the public key of A, not further shown here). Note that we assume here that B's identity cannot be recovered by inspecting the cellular network, see section 3.4.

– The authorities A store this information in their database, indexed using the public key K_C used to encrypt it. Information older than Δ days should automatically be pruned (as this information can never be decrypted anymore, see below).

In this protocol, the log $\mathscr{L}_C[]$ contains all private/public key pairs k_C, K_C that C broadcast and that have been used by all phones in the vicinity of C the last Δ days to encrypt their identities id_{B_i} before submitting them to the authorities. When someone is diagnosed as being infected, *only the public keys* in the log are extracted from their phone C (similar to the previous protocol), and sent to the authorities. This allows the authorities to search for any entries $E_{K_C}(E_{K_A}(id_B) \,\|\, e_i \,\|\, h(e_{i-1}))$ stored in their database indexed by the public key K_C. All entries found are sent to device C which uses the corresponding private keys k_C stored in the log to decrypt.

As a result, device C obtains entries $E_{K_A}(id_B) \,\|\, e_i \,\|\, h(e_{i-1})$ for all devices B that have been in close contact with C. To determine whether this contact was sufficiently long, C looks for any two entries $E_{K_A}(id_B) \,\|\, e_i \,\|\, h_i$ and $E_{K_A}(id_B) \,\|\, e_j \,\|\, h_j$ (collected for the private and public keys used on the same day d) such that $h_i = h(e_j)$ (or vice versa of course). The precision of this approach can be improved if one makes each epoch shorter, and requires the matching process to find a sufficiently long chain of epochs that together cover at least T seconds (the minimum time before a contact is deemed sufficiently risky). For any such entries found it returns $E_{K_A}(id_B)$ to the authorities. Using their knowledge of k_A they can decrypt this to recover id_B.

B itself ensures that it only reports its identity if it sees another device C to be closer than the unsafe distance. B relies on C to only return $E_{K_A}(id_B)$ when this encrypted identity is found to be linked to consecutive epochs, and not to leak information about the time and the actual length of the encounter (which C could deduce based on its knowledge of when it used a particular key, and using repeated chaining of epoch hashes).

5.1 Analysis

Let DB be the database of all entries $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_i \,\|\, h(e_{i-1}))$ stored by the authorities. In this protocol $\mathsf{contacts}_X$ is defined to be all Y such that DB contains two entries $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_i \,\|\, h_i)$ and $E_{K_{X'}}(E_{K_A}(id_Y) \,\|\, e_j \,\|\, h_j)$ encrypted using two public keys K_X and $K_{X'}$ used by device X on the same day such that $h_i = h(e_j)$ (or vice versa of course).

We again analyse the requirements outlined in section 3 one by one.

Completeness We have to show that if $\mathsf{contact}(X, Y)$ then $Y \in \mathsf{contacts}_X$, and that whenever device X becomes infected, the health authorities A learn id_X and all id_Y such that $Y \in \mathsf{contacts}_X$.

The first part is shown as follows. Suppose $\mathsf{contact}(X, Y)$. Then there is a time t such that X and Y have been within distance d of each other for at least $T + 2\delta$ time. Time is divided into epochs ϵ_i that start at time $i \cdot T$ and last for T seconds. As X broadcasts its current public key K_X at least every δ seconds, this means Y must receive X's current public key in at least two consecutive epochs e_{i-1} and e_i with sufficient signal strength. This means Y sends $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_{i-1} \,\|\, h(e_{i-2}))$ and later $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_i \,\|\, h(e_{i-1}))$ to the authorities. X's key may have changed during this period. Regardless, the conditions for $Y \in \mathsf{contacts}_X$ are satisfied.

The second part follows from the fact that once X is diagnosed as infected, it sends its public keys K_X in its log to the authorities, who use it to look up all entries $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_i \,\|\, h(e_{i-1}))$ in DB. These are sent to X for decryption. X can decrypt these as long as they are not older than Δ days, because that is how long X keeps the private keys k_X it used in its logs. We already established that both $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_{i-1} \,\|\, h(e_{i-2}))$ and $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_i \,\|\, h(e_{i-1}))$ are in DB. This means X obtains $E_{K_A}(id_Y) \,\|\, e_{i-1} \,\|\, h$ and $E_{K_A}(id_Y) \,\|\, e_i \,\|\, h$. As $h(e_{i-1}) = h'$ we have a match and X returns $E_{K_A}(id_Y)$ to the authorities, who can now decrypt it to recover id_Y as required. They also learn id_X as soon as X tests positive.

Precision/soundness We have to show that the health authorities A only learn id_Y when Y itself becomes infected or when $\mathsf{contact}(X, Y)$ for an infected device X (within the last Δ days measured from the moment user X tested positive and consented to sharing information).

First observe that according to the protocol, only infected devices X send their identifier to the authorities directly. Other devices Y only send $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_i \,\|\, h(e_{i-1}))$ whenever they receive a public key from another device X that they detect to be within unsafe distance d. These messages cannot be decrypted after Δ days because X purges keys older than that from its log.

As we have seen above, the authorities need the help of an infected device X to decrypt such messages. If not $\mathsf{contact}(X, Y)$, it is unlikely (though not impossible if X and Y are close around an epoch rollover) that Y sent two messages $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_{i-1} \,\|\, h(e_{i-2}))$ and $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_i \,\|\, h(e_{i-1}))$ for two consecutive epochs, which would trigger X's rule to return $E_{K_A}(id_Y)$ to the authorities. Again precision only holds approximately.

Confidentiality of infection We have to show that only the health authorities A learn which devices are infected.

This immediately follows from the fact only when someone is diagnosed as being infected, their identity id_X is extracted from their phone X and sent to the authorities, and that no information (about infection status) leaves the authorities. Throughout the protocol the identity of a device is always encrypted against the key of the authorities.

Note that the authorities *do* learn how often (over the full period of Δ days) an infected device X was in close contact with another device Y (but not on which exact time or on which days in particular).

Confidentiality of infected contact Suppose X is infected. We have to show that, apart from the health authorities A, no entity learns about a device Y such that $Y \in \mathsf{contacts}_X$.

Note that X (only) receives entries $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_i \,\|\, h(e_{i-1}))$ which it can decrypt to obtain $E_{K_A}(id_Y) \,\|\, e_i \,\|\, h(e_{i-1})$. As $E_{K_A}(id_Y)$ is semantically secure, this does not leak information about id_Y (or the frequency with which id_Y occurs in the entries that X receives). Again, throughout the protocol the identity of a device is always encrypted against the key of the authorities.

Confidentiality of uninfected contacts Suppose X is not infected. We have to show that no entity learns about a device Y such that $Y \in \mathsf{contacts}_X$.

This trivially follows from the fact that in that case X is not asked to match any data on behalf of the authorities. Then the authorities store $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_i \,\|\, h(e_{i-1}))$ safely encrypted against both K_A and K_C. Again, throughout the protocol the identity of a device is always encrypted against the key of the authorities.

Location privacy We have to show that an adversary cannot track a device while it moves around from one physical location to another.

A device X either broadcasts a random key K_X, a signed timestamp $[t]_{k_X}$ or an encrypted identifier $E_{K_X}(E_{K_A}(id_Y) \,\|\, e_i \,\|\, h(e_{i-1}))$

For the random public key, we again assume it is refreshed whenever the MAC address of the Bluetooth channel changes. The encrypted identifier does not pose an issue as the message is encrypted using a semantically secure cipher. In both cases, a new message can never be linked to the previous message from the same device. The signature is always different because the time changes. When clocks are perfectly synchronised, and change MAC address at the very same time, two different devices both send the same signature. If however, clocks are not perfectly synchronised and MAC addresses do not change at exactly the same time, then two devices could be traced

because one device emits a sequence $[t]_{k_X}, [t+\omega]_{k_X}, , [t+2\omega]_{k_X}, \ldots$ while the other emits a sequence $[t']_{k_X}, [t'+\omega]_{k_X}, , [t'+2\omega]_{k_X}, \ldots$

Authenticity We have to show that a set of adversaries $\mathscr{M}$ cannot convince a device X that $Y \in \mathsf{contacts}_X$ for some device Y for which there is no t such that $X \approx_t Y$, unless there are $M, M' \in \mathscr{M}$ such that both $X \approx_t M$ and $M' \approx_t Y$ for some time t.

This follows from the fact that Y never sends a message to the authorities encrypted against a public key K_X that wasn't recently (according to the signature $[t]_{k_X}$) sent by X itself. This means that either X is close to Y, or its messages are relayed in real time by some adversarial devices.

6 Related work

Many papers have been published last year proposing some protocol for contact tracing or exposure notification. Martin *et al.* [22] offer an excellent overview of the state of the art. We highlight the main Bluetooth-based approaches here and compare them with our proposal.

One of the first protocols of these type was the TraceTogether app deployed in Singapore [21]. This inherently centralised approach lets phones exchange regularly changing pseudonyms over Bluetooth. A phone of a person who tests positive is requested to submit all pseudonyms it collected to the central server of the health authorities, who are able to recover phone numbers and identities from these pseudonyms. This allows them to quickly contact anybody in close contact with this infected person. Privacy is clearly less of a concern. [22]

Other notable early protocols are Cho *et al.* [5] (that use an anonymous message passing approach to allow devices to send infection status messages to other devices that were in close proximity), and Canetti *et al.* [3] that is very similar to the DP-3T to be discussed next.

6.1 The DP-3T protocol

The Decentralized Privacy-Preserving Proximity Tracing (DP-3T) protocol[23] is a decentralised protocol for exposure notification [31]. The protocol uses

[21] https://www.tracetogether.gov.sg

[22] The Singapore authorities recently announced that the police can access COVID-19 contact tracing data for criminal investigations. https://www.zdnet.com/article/singapore-police-can-access-covid-19-contact-tracing-data-for-criminal-investigations/

[23] https://github.com/DP-3T/documents

locally generated frequently changing ephemeral identifiers (EphIDs) that devices broadcast via Bluetooth Low Energy advertisements. Other devices store the EphIDs they observe, together with the duration and a coarse indication of the time of contact. In the so called "Low-Cost" design of DP-3T, the device of an infected user uploads all EphIDs *it itself generated* to a central server. Other devices regularly query this server to download any new EPhIDs and locally match these with EphIDs received earlier from devices in close proximity. Any match indicates a contact, of which the user is notified by the app itself. The central server does not learn these matches. A variant of this protocol uses Cuckoo hashing to hide the actual EphIDs of infected users, offering only the resulting Cuckoo filter for download to other devices that want to check whether they have been in close contact with an infected user. This is done to somewhat mitigate the otherwise existing risk that the EphIDs of infected users revealed can be used to retroactively track their location [33].

The DP-3T consortium has done a tremendous amount of work in an incredible short amount of time by creating specifications [31], reference implementations[24], and a detailed risk analysis [30]. Their efforts, also influencing the policy agenda of the European Commission and beyond, are a shining example for all of us working in the area of privacy enhancing technologies. The short description given here barely does their work justice.

Google and Apple released an interoperable framework for exposure notification (GAEN) based on the DP3T protocols. Elsewhere we argue that this creates a dormant functionality for mass surveillance at the operating system layer, and show how it does *not* technically prevent the health authorities from implementing a purely centralised form of contact tracing (even though that is the stated aim) [17].

The main advantage of the protocols discussed here (over DP3T and GAEN) is that information revealed by infected users cannot be used to retroactively track their location. Moreover, our protocols solve contact tracing, not exposure notification.

6.2 The DESIRE protocol

The DESIRE protocol[25] (designed by INRIA PRIVATICS TEAM as a a follow-up to their ROBERT protocol[26]) is a hybrid solution for contact tracing where determining the risk of infection happens centrally. Instead of collecting

[24] https://github.com/DP-3T/
[25] https://github.com/3rd-ways-for-EU-exposure-notification/project-DESIRE
[26] https://github.com/ROBERT-proximity-tracing

(and sharing) temporary pseudonyms (as e.g., the DP3T protocol does), DESIRE is based on Private Encounter Tokens (PETs) instead, that remove the tracking risks inherent to decentralised solutions based on pseudonyms, and that also mitigates against replay attacks. These PETs are essentially Diffie-Hellman based shared secrets generated as follows. Devices A and B regularly generate a new device secret s_A and s_B and use that to generate Ephemeral Bluetooth Identifiers (EBID) $E_A = g^{s_A}$ and $E_B = g^{s_B}$ (within some suitable group with generator g). Devices broadcast these EBIDs using Bluetooth. When receiving an EBID E_A, device B computes a Private Encounter Tokens $P_{BA} = H(E_A^{s_B}) = H(g^{s_A s_B})$. Device A similarly computes $P_{AB} = P_{BA}$ when receiving E_B. Devices of infected users upload all PETs they collected to the central server. Devices of other users regularly upload a list of collected PETs they collected to allow the central server to compute a risk score.

This means that DESIRE is different from our protocols in that it requires the active participation of devices that have been in contact of an infected device not only during the moment of contact itself, but also *post-contact* after the infected device was notified of being infected. DESIRE also assumes synchronised clocks (which our first protocol does not rely on).

DESIRE also requires a handshake (i.e., a successful bidirectional exchange of messages between two proximate devices) for a contact to be successfully registered. Our second protocol thus also improves on the DESIRE protocol in this respect. Our protocols allow similar techniques to be deployed as described in the DESIRE proposal to prevent the authorities to learn which log entries belong to which particular infected user (thus hiding even the social graph of infected users from the authorities). For example, log entries from several devices could be uploaded simultaneously through a mixing network [4].

7 Conclusion

We have proposed two centralised protocols for digital contact tracing that, contrary to the common hypothesis that this is an inherent risk, do not allow (retroactive) tracking of the location of a device over time. We have done so even though we have strong reservations against the use of contact tracing in fighting the epidemic (given the significant privacy infringements it causes and the limited effectiveness to expect of it), as we expect governments to implement such systems for contact tracing anyway. In that case, it is important to have relatively comprehensive map of the design space available to guide them in their choice.

We thank the anonymous referees for their very insightful comments that helped to improve the protocols and the presentation.

References

[1] J. Bay. “Automated contact tracing is not a coronavirus panacea”. *Medium* (Apr. 11, 2020).

[2] D. J. Bernstein. “Curve25519: New Diffie-Hellman Speed Records”. In: *Public Key Cryptography - PKC 2006* (New York, NY, USA, Apr. 24–26, 2004). LNCS 3958. Springer, 2004, pp. 207–228.

[3] R. Canetti, A. Trachtenberg, and M. Varia. “Anonymous Collocation Discovery: Harnessing Privacy to Tame the Coronavirus”. *CoRR* abs/2003.13670 (2020). arXiv: 2003.13670 [cs.CY].

[4] D. Chaum. “Untraceable electronic mail, return addresses, and digital pseudonyms”. *CACM* 24.2 (1981), pp. 84–88.

[5] H. Cho, D. Ippolito, and Y. W. Yu. “Contact Tracing Mobile Apps for COVID-19: Privacy Considerations and Related Trade-offs”. *CoRR* abs/2003.11511 (2020). arXiv: 2003.11511 [cs.CR].

[6] Y.-A. de Montjoye, F. Houssiau, A. Gadotti, and F. Guepin. “Evaluating COVID-19 contact tracing apps? Here are 8 privacy questions we think you should ask.” Computational Privacy Group Blog. Apr. 2, 2020.

[7] P.-O. Dehaye. “Inferring distance from Bluetooth signal strength: a deep dive”. *Medium* (May 19, 2020).

[8] R. Dingledine, N. Mathewson, and P. F. Syverson. “Tor: The Second-Generation Onion Router”. In: *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*. USENIX, 2004, pp. 303–320.

[9] ECRYPT. *D5.4 - Algorithms, Key Size and Protocols Report (2018)*. Report. ECRYPT – CSA, Feb. 28, 2018.

[10] G. Eyal. “Beware the Trolley Zealots”. *Sociologica* 14.1 (2020).

[11] L. Ferretti et al. “Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing”. *Science* (Mar. 31, 2020).

[12] FIPS 202. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. NIST FIPS PUB 202. NIST, Aug. 2015.

[13] M. Foucault. “Omnes et Singulatim: Towards a Criticism of Political Reason”. In: *The Tanner Lectures on Human Values*. Ed. by S. McMurrin. Vol. II. Delivered at Stanford University, October 10 and 16, 1979. Salt Lake City: University of Utah Press, 1981, pp. 225–254.

[14] P. Fouque, A. Joux, and M. Tibouchi. “Injective Encodings to Elliptic Curves”. In: *Information Security and Privacy - 18th Australasian Conference, ACISP 2013, Brisbane, Australia, July 1-3, 2013. Proceedings*. Vol. 7959. Lecture Notes in Computer Science. Springer, 2013, pp. 203–218.

[15] B. Golder. “Foucault and the Genealogy of Pastoral Power”. *Radical Philosophy Review* 10.2 (2007), pp. 157–176.

[16] R. Hinch et al. “Effective Configurations of a Digital Contact Tracing App: A report to NHSX”. Apr. 16, 2020.

[17] J.-H. Hoepman. “A Critique of the Google Apple Exposure Notification (GAEN) Framework”. CoRR abs/2012.05097. Dec. 2020. arXiv: 2012.05097 [cs.CY].

[18] P. Howell O’Neill. “No, coronavirus apps don’t need 60% adoption to be effective”. *MIT Technology Review* (June 5, 2020).

[19] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition*. 2nd. Boca Raton: CRC Press, 2015.

[20] A. Langley, M. Hamburg, and S. Turner. *Elliptic Curves for Security*. RFC 7748. RFC Editor, Jan. 2016, pp. 1–22.

[21] J. Martin et al. “A Study of MAC Address Randomization in Mobile Devices and When it Fails”. *PoPETs* 2017.4 (2017), pp. 365–383.

[22] T. Martin, G. Karopoulos, J. L. Hernandez Ramos, G. Kampourakis, and I. Nai Fovino. “Demystifying COVID-19 digital contact tracing: A survey on frameworks and mobile apps”. *Wireless Communications and Mobile Computing* (2020), p. 8851429.

[23] S. McDonald. “The Digital Response to the Outbreak of COVID-19”. *Centre for International Governance Innovation online* (Mar. 30, 2020).

[24] Y.-A. de Montjoye, F. Houssiau, P. Sapieżyński, and L. Radaelli. “Cambridge Analytica is only the beginning and you might have your friends to blame for it”. Computational Privacy Group Blog. Mar. 29, 2018.

[25] J. Morley, J. Cowls, M. Taddeo, and L. Floridi. “Ethical guidelines for COVID-19 tracing apps. Protect privacy, equality and fairness in digital contact tracing with these key questions.” *Nature* 582.29–31 (May 28, 2020).

[26] E. Morozov. *To Save Everything, Click Here. The Folly of Technological Solutionism*. New York: PublicAffairs, 2013.

[27] P. Mozur, R. Zhong, and A. Krolik. “In Coronavirus Fight, China Gives Citizens a Color Code, With Red Flags”. *The New York Times* (Mar. 1, 2020).

[28] E. Redmiles. “How to Fix COVID Contact Tracing”. *Scientific American* (Dec. 7, 2020).

[29] L. Taylor, G. Sharma, A. Martin, and S. Jameson. *Data Justice and COVID-19: Global Perspectives*. London: Meatspace Press, 2020.

[30] The DP-3T Project. *Privacy and Security Risk Evaluation of Digital Proximity Tracing Systems*. Whitepaper. DP-3T Consortium, Apr. 19, 2020.

[31] C. Troncoso et. al. *Decentralized Privacy-Preserving Proximity Tracing*. Whitepaper. DP-3T Consortium, May 25, 2020.

[32] Y. Tsiounis and M. Yung. “On the Security of ElGamal Based Encryption”. In: *Public Key Cryptography, First International Workshop on Practice and Theory in Public Key Cryptography, PKC ’98, Pacifico Yokohama, Japan, February 5-6, 1998, Proceedings*. Vol. 1431. Lecture Notes in Computer Science. Springer, 1998, pp. 117–134.

[33] S. Vaudenay. “Analysis of DP3T”. *Cryptology ePrint Archive* 2020/399 (2020).

[34] E. Verheul and B. Jacobs. “Polymorphic Encryption and Pseudonymisation in Identity Management and Medical Research”. *Nieuw Archief voor Wiskunde NAW*. 5th ser. 18.3 (2017), pp. 168–172.

[35] L. White and P. van Basshuysen. “Without a trace: Why did corona apps fail?” *Journal of Medical Ethics* (2020). Epub ahead of print: [2021-01-12].

[36] World Health Organization. “Contact tracing in the context of COVID-19, Interim guidance”. May 10, 2020.