Security and privacy analysis of the document 'ROBERT: ROBust and privacy-presERving proximity Tracing'

The DP-3T Project

22 April 2020

On 18 April 2020 Inria[1] and Fraunhofer AISEC released a document on their GitHub repository proposing ROBERT as a candidate for PEPP-PT contact tracing (CT).[2] In this document, the DP-3T project provides an analysis of this system to inform open discussion. We note that many attacks are common to the implementation of the PEPP-PT Data Protection and Security Architecture that will be deployed in Germany.[3]

This approach differs from DP-3T in several fundamental ways, notably, an infected user sends observed Bluetooth ephemeral identifiers to the server, rather than the user's own emitted identifiers, and centralizes the risk calculation at the server, rather than performing it locally on individual smartphones.

Overview of ROBERT

The contact tracing protocol ROBERT works in 4 phases[4]:

1. **Installation**. When the app is installed on a smartphone, it registers with the backend server. This registration results, on the server, in a permanent identifier (ID), a symmetric shared key K_A and a number of ephemeral Bluetooth identifiers (EBIDs) that are pushed to the phone to be broadcast. EBIDs are generated by the server by encrypting the ID with a server key K_S. New EBIDs are pushed to a phone by the server periodically after installation.
2. **Normal operation**. Each app broadcasts EBIDs, in so-called *Hello* messages, via Bluetooth and records EBIDs seen in the broadcasts of other phones in close physical proximity.
3. **Handling infected patients**. After a patient is diagnosed, and with their consent and an authorization from a health authority, the app ***uploads all recorded EBIDs*** observed *during the past three weeks*. This upload is done using an anonymous communication channel to ensure that the server cannot link different EBIDs to the same sender. Along with each EBID, the phone sends the time of contact. The server decrypts the EBIDs with its key to recover the permanent ID, and flags as "exposed" any of the permanent IDs for which at least one EBID has been seen.

[1] An amended version of the document was uploaded on 19th April to specify that authorship is that of the PRIVATICS team.
[2] ROBERT Spec (18.4.2020) retrieved https://github.com/ROBERT-proximity-tracing/documents on 18.04.2020.
[3] PEPP-PT, *Data Protection and Security Architecture: Illustrated on the German Implementation* (18 April 2020). Retrieved from https://github.com/pepp-pt/pepp-pt-documentation/ on 18.04.20.
[4] For simplicity we omit the country code from the description.

4. **Exposure status request**. Periodically, the app on each phone checks with the server if it has been exposed to infected users (i.e., whether any of its EBIDs have been observed by an infected person).

This protocol explicitly seeks to provide accuracy and reliability of proximity data, anonymity of users from other users, and anonymity of users from a central authority whilst working under the following assumptions:

- Users are malicious: they can eavesdrop and modify the protocol, inject/modify messages, modify the app's code, etc.
- The backend server is honest-but-curious: it will not actively try to modify its operation, spy on, or de-anonymize people; it will not collude with others; etc. However, it will try to learn as much as possible from the information it receives and stores.

Given the fluidity of the current policy environment, it is unclear that these are appropriate assumptions to make. In particular, a significant concern that we will address below is function creep: the possibility that this system could be adapted for surveillance purposes (far) beyond its initial use. **We feel it is important to give the public a clear understanding of how easily a system of this type can be repurposed to perform intrusive surveillance and how our DP-3T proposal corrects these privacy flaws without compromising the CT functionality.** (We also note that some of the terminology in the specification of ROBERT suggests that the backend server is completely trusted)[5].

Privacy analysis

The ROBERT design raises the following privacy concerns:

Tracing of users

As the backend server creates the ephemeral identifiers, the backend can, at any point, link the past and future ephemeral identities of *any user, infected or not*, by decrypting back to their permanent identifier. This ability puts the backend server in a position of power with ***high potential for function creep***. Function creep is a highly relevant discussion in the current pandemic, since a tool designed for one task, if function creep is technically possible, can easily be turned into an instrument of surveillance with considerable human rights implications.

First, the protocol enables **tracing of all individuals across time.** The backend can convert any EBID into a permanent identifier, regardless of whether that EBID belongs to an infected person or not. As a result, the backend can associate any observation -- reported by an infected person, or reported by any Bluetooth sensor -- to one specific pseudonymous individual. In combination with other datasets, such as a use of a registered 'smart' travel card or CCTV footage, the server can, at scale, connect

[5] See also issue #13 on ROBERT GitHub: https://github.com/ROBERT-proximity-tracing/documents/issues/13

pseudonyms to real identities, and track all individuals, infected or not in space, across the EBIDs generated from their PUID.

Second, given a target EBID[6] (e.g., collected by law enforcement from a suspect, or at a passport control point), it is possible to **tag and classify individuals that third parties can recognise without access to the server database**. Recall that the backend can convert any EBID into a permanent identifier. Therefore, given a target EBID, the backend can provide *any* EBID to the corresponding user. (Note that EBIDs are not authenticated and the server does not provide any proof that they are an encryption of the ID or that the correct key was used.) This capability can enable law enforcement, or other actors, to trace, *without access to the backend database*, the movements of users and communities by assigning them distinguishable identifiers and recognizing their tagged Bluetooth emissions.

- **Deanonymization using specific EBIDs**. The above technique can easily de-anonymize a target by providing them with distinguishable EBIDs that make them recognizable.

- **Long term persistent tracking of individuals**. In the extreme, the backend could assign special keys to certain users (note that a user has no way to know whether their key changes over time or not) and leak them selectively, enabling long-term tracking by third parties. We note that this also works for communities, as one could assign specific identifiers to target groups of people.

Learning infected close contacts

A tech-savvy user who is notified by an app that they have been in contact with an infected user can identify this individual. This is a fundamental challenge to <u>all</u> contract tracing systems, including those based on ROBERT and DP3T. The procedure is simple:

1. The tech-savvy user creates a number of accounts, one for every slot of the day in which they may want to identify an infector. For instance, to split the day in slots of 15 minutes, the user needs 96 accounts. Since the ROBERT proof of work has to be simple enough to enable creation of individual accounts fairly quickly, we assume that 96 accounts can be created in a reasonable amount of time[7], even faster if commodity cloud computing is used.

2. The tech-savvy user rotates between the different EBIDs of these accounts during the day and maintains a record of which account was used in each time slot, the EBIDs of observed users, and the actual identities (photos would suffice) of the people he came into contact with during each slots.

[6] We are aware that the assumptions of ROBERT indicate that the backend will not collude with other entities. We however believe that this belief is unrealistic, as being able to assign identifiers to targets provides law enforcement with targeted tracking means that they do not currently have.
[7] If the Proof-of-work is the same as the German implementation, NTK, this time is 3 hours.

3. When the tech-savvy user receives a notification that he is at risk, he uses the specific account to identify the slot of the day in which the contact with the infected person happened. From his records, the tech-savvy user can identify the infector.

Note that this attack does *not* depend on the information saved on the server, only on the local information of the adversary. Therefore it cannot be avoided. In fact, this attack is possible in any contact tracing system (including DP3T).

We note that ROBERT suggests mitigating an attack similar to this one by introducing a probabilistic element whereby there is a random chance (the authors suggest 5% or 10%) that a user who should not be considered epidemiologically at risk is given instructions as if they were. This could create problems of adherence to guidance if users use such deniability to argue they are not at risk, test capacity issues if they take the guidance seriously, and civil liberties challenges if the warning is tied to a coercive outcome such as quarantine. Moreover, we note that any potential deniability can be mitigated easily in practice by using multiple devices or accounts simultaneously to increase certainty about their result.

Linkability attacks

Upon infection, the backend server receives the local contact list of the infected user. Contacts are uploaded one at a time, in a way so that they are not linkable (see discussion below on anonymizing channels). Such a contact list does not contain the identifiers of the uploader (i.e., does not reveal the infected person to the backend server). It only includes the received "Hello" messages of the contacts made by the infected person along with the timestamp of reception. We discuss attacks to recreate these missing links (between infected uploaders and their contacts), as well as what other links can be inferred through this data (e.g., co-location between two non-infected users).[8,9]

- **The server sees permanent IDs.** We recall that even though the Hello packets contain ephemeral EBIDs which vary per epoch and per user, the backend server is able to map them back to their respective permanent IDs.

- **Timestamp-based linkability.** The Hello messages uploaded contain a timestamp of reception. If the timestamps sent to the backend server are precise enough (which we can assume, as the check for replay attacks is made in seconds, page 10 of the ROBERT proposal), this could lead to timestamp-based intersection attacks that allow the backend to infer that two infected users (i.e., users who uploaded their contact lists to the backend server) were in contact at some point in time. This enables the server to build a partial social graph that reflects the encounters

[8] See also issue 11 in ROBERT Github: https://github.com/ROBERT-proximity-tracing/documents/issues/11

[9] We are aware that the assumptions of ROBERT indicate that the backend will not try to de-anonymize or link. We believe that this belief is unrealistic, as it cannot be ensured, and therefore analyze its consequences.

between infected people. As a large percentage of the population will eventually get infected, we expect this graph to be fairly large.

Consider Alice who anonymously uploads a contact with Bob's long-term ID at a timestamp X. Later on, Bob anonymously uploads the same contact with Alice's long-term ID, also at timestamp X± Δ where Δ is very small (e.g., milliseconds). In theory, the server should only see two unlinkable events since both Alice and Bob were anonymous (it might have been that Claire and Denis made the reports), but as the timestamps are very close in time, the server can infer that Alice and Bob are two infected uploaders.

- **Timestamp-based co-locations.** This time-stamp based linkability can also leak co-locations of [infected, non-infected] users or two non-infected users. For instance, consider Alice, Bob who are sick and upload their contacts; Alice uploads [Bob, Charlie] and Bob uploads [Alice, David], both for epoch T. Potentially, the timestamps will reveal that Alice and Bob were in contact as described above; if so, it means that Alice, Bob, Charlie and David were all co-located at the time of contact.

- **Backend Server Might Learn Co-locations (even without precise time-stamps).** As a toy example, imagine that for an epoch T only two uploads are done. If Alice, Bob, and Claire had contacts at time T, and both Alice and Bob end up sick and upload their contact lists [(Claire, T), (Bob, T)] and [(Claire, T), (Alice, T)] respectively, the server will see the following contacts at time T: [Bob, 2x Claire, Alice]. It learns that [Alice, Bob, Claire] were co-located during this epoch. It also learns that Claire is not an uploader (more on that below).

  Naturally, outside of this toy example, the backend will have a (very) partial or incomplete view of the real co-locations due to missing links; but each inferred co-location will be correct.

- **Identifying anonymous uploaders with co-locations.** Imagine the following toy example. Suppose Alice, Bob, and Claire meet at a time epoch T. Their contact lists are:

  Alice: [..., (Bob, T), (Claire, T), ...]
  Bob: [..., (Alice, T), (Claire, T), ...]
  Claire: [..., (Alice, T), (Bob, T), ...]

  Alice gets infected and uploads her contact list. The server has the following information in its database:

(Bob, T)
(Claire, T)
...

Later on Bob gets infected. The server has the following information in its database:

(Bob, T)
(Claire, T)
...
(Alice, T)
(Claire, T)
...

Doing an intersection at time t and a frequency analysis the server finds that:

(Alice, T) --> 1x
(Bob, T) --> 1x
(Claire, T) --> 2x

At this stage, the backend server knows there are at least 2 infected people and at most 5 people involved in the social graph at time T (Alice, Bob, Claire, and potentially two unknown infectors).

In this example, Claire cannot be an infector since her tuple appears twice (also assuming that devices upload unique Hello messages and not how many times they might have received them).

This toy example further assumes that Alice, Claire, Bob are the only contacts in the database at time T, which naturally allows the system to infer that Alice and Bob are infected. In practice, the infectors could be two other random persons who were near to Alice, Bob and Claire, but who somehow do not appear in the contacts of either of them. This seems less likely; the partial data clearly suggests that Alice and Bob are the infectors despite the uncertainty, and we note that the uncertainty drops as the graph gets denser (with a larger percentage of the population getting infected). More fundamentally, the system does not provide any mechanism or guarantee to ensure this scenario does not happen (e.g., enforcing a minimum number of contacts per epoch).

- **Identifying anonymous uploaders with causality.** Causality in the upload is preserved. In short, when Alice infects Bob, the time-separation between the two sequences of uploads of Alice and Bob will have to be much larger than any total random delay added to each upload. Therefore, the honest-but-curious backend will not be confused between the uploads of Alice and Bob despite added random delays.

  Consider the following example, illustrated in Figure 1: say Alice is sick (but does not know it), she infects Bob at time T1, then she is diagnosed as infected and

uploads her contact list at time T2. As a consequence, Bob becomes sick after the incubation period (which is a few days), then uploads his contacts at time T3. In fact, to preserve the utility of the system, Alice's messages **must** be uploaded at T2<<T3 so there is a possibility to contact Bob and tell him that he is at-risk/sick. If Alice delays her messages to hide causality, the system would miss notifying at-risk people.

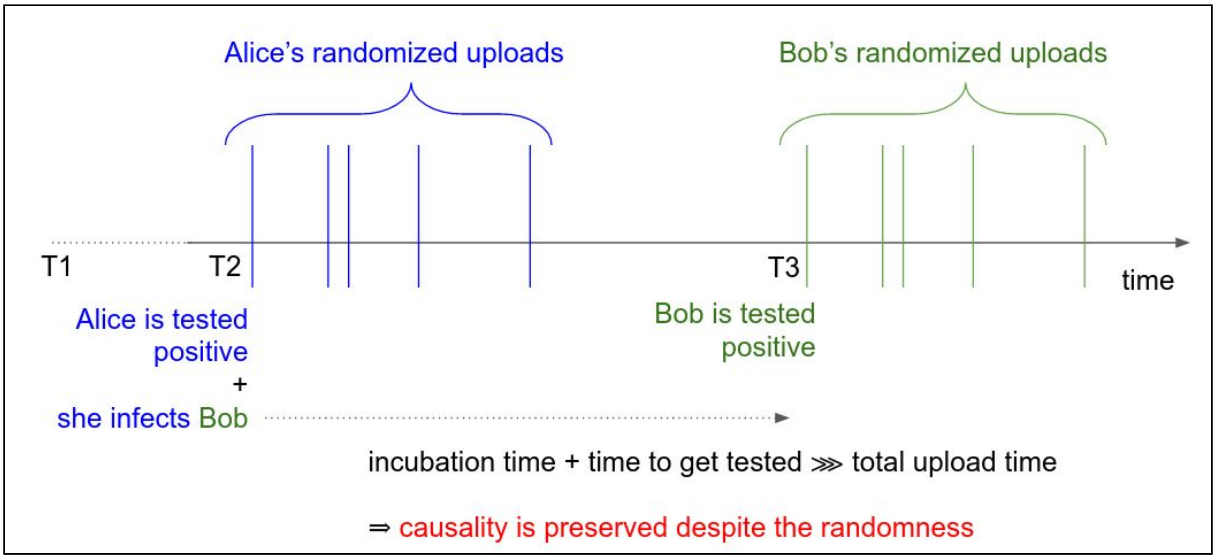

Figure 1

Both "anonymous" upload methods (uploading contacts one-by-one with randomized timings, or having the healthcare provider group uploads from several users) preserve causality: when Alice infects Bob, the server will see all of Alice's contacts long before seeing Bob's contacts.

Therefore, the server can reconstruct a pseudonymous graph (whose vertices are a **set** of possible users and edges are the infections) using time causality.

Consider the following example (illustrated below in Figure 2). Say Denis is sick, he infects Alice at time T1, then uploads his observed EBIDs. As a consequence, Alice becomes sick after the incubation period (or she receives an at-risk notification), then she uploads her contacts at time T2 >> T1 + Total time for Denis to upload his contacts.

The server sees the following contacts at time T1: [Alice, Bob, Claire], (... after a few days ...) [Bob, Claire, Denis]. As said in the previous attack, the server learns (1) [Alice, Bob, Claire, Denis] were co-located during this short epoch T, and (2) that none of the uploaders is Claire or Bob. In addition, using the time-difference between the uploads, it learns (3) it is Denis that infected Alice. We note that it could be two other unknown participants if they also both happen to not have had contact with Alice and Bob; we expect this scenario to become less likely as many people become infected (and the graph becomes denser).

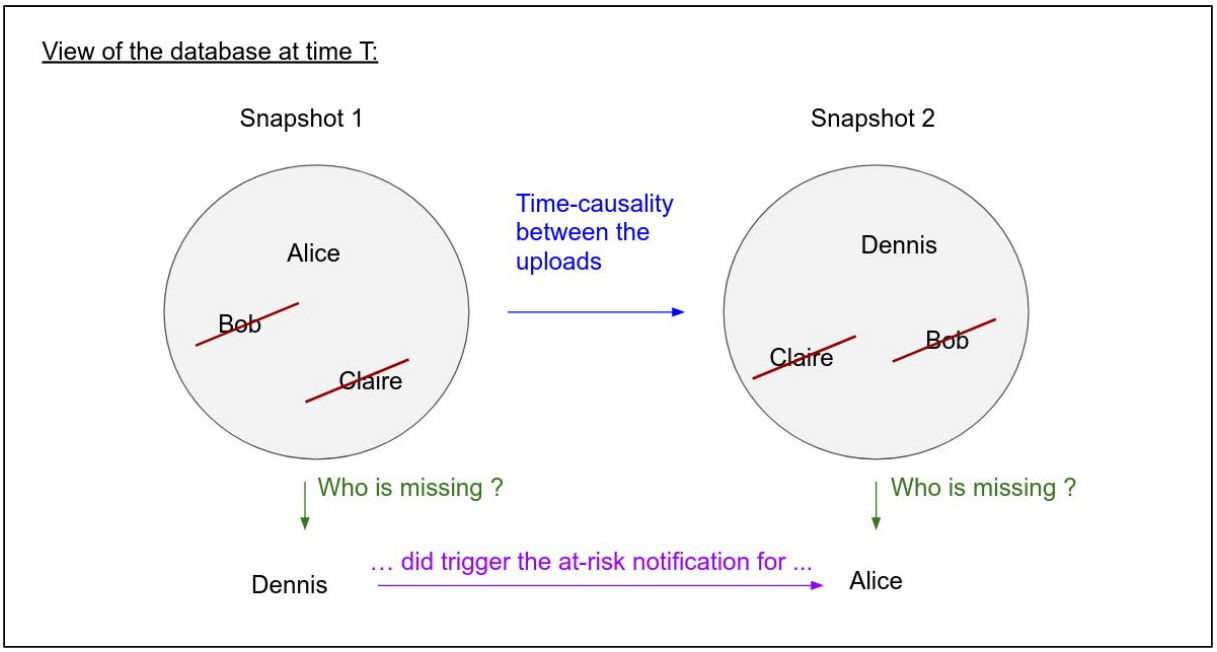

Figure 2

- **Identifying anonymous uploaders with frequency analysis and causality:** We can generalize the previous attack using frequency analysis. In general, we know that contacts that appear the **same number of times** in both sets cannot be uploaders. If they appear a different number of times, we know it is possible that they were an uploader in a particular set (or that Bluetooth contact-tracing didn't work well). If the graph is dense, epochs are short, and (Bluetooth) contact-tracing works well, then frequency analysis will likely reveal the most likely infectors/infected relationships. This could allow reconstructing a graph of social interactions.

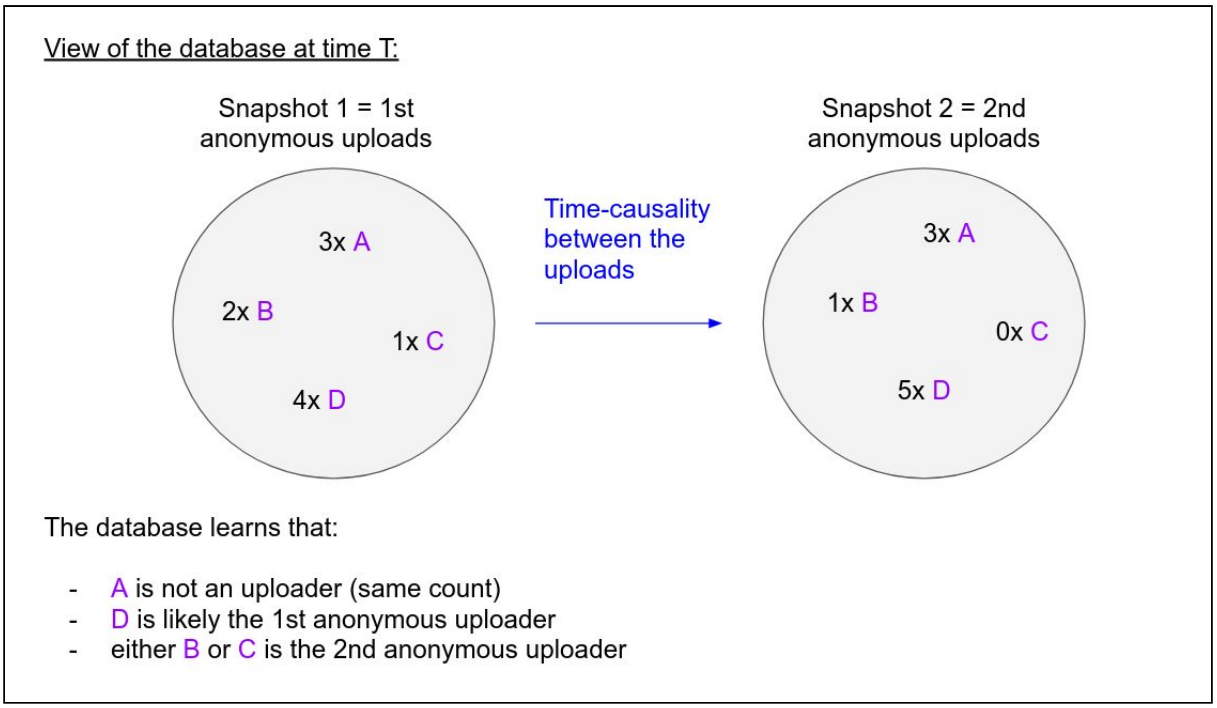

Figure 3

The information collected in the backend **enables the honest-but-curious server to run several linkability attacks**, *e.g.,* identifying an otherwise anonymous uploader, inferring co-locations of infected and non-infected users, inferring social relationships, etc. **This can be done with only information that is currently collected by the backend**. It is difficult to quantify the extent of this kind of attacks; their success is dependent on timestamp granularity, particular social/contact patterns, and potential auxiliary knowledge that the backend server might obtain. We acknowledge that we demonstrate attacks in corner cases. **However, the system has no mechanism to ensure such attacks will not be possible** (even though the information is collected in a careful manner, e.g., by uploading contacts one-by-one or by uploading the aggregated contacts of multiple infected users) and we expect these attacks to still work on larger datasets.

We emphasize that these conclusions are made in the honest-but-curious backend model which does *not* obtain information from ISPs (which are considered honest) and network eavesdroppers (which are not considered). Naturally, if an ISP is also honest-but-curious and shares information with the backend, they can perform similar attacks much more easily and more accurately.

On the anonymity of individual uploads

Network Address Translators (NATs) as an anonymization mechanism. A 2016 academic study reported that more than 90% of mobile operators deployed carrier-grade NATs[10], with a higher deployment rate in Asia and Europe necessitated by the shortage of available IP blocks in these regions. While this study shows that the use of carrier-grade NATs is prevalent in mobile networks, it is incorrect to assume that every single mobile user accesses the Internet through a NAT, particularly with an increasing number of mobile ISPs deploying IPv6 networks. Furthermore, one cannot assume that the mobile device will be constantly accessing the network over a cellular link as they roam between WiFi and cellular networks.

Moreover, even if NATs are prevalent, they typically group users in one location, resulting in disjoint anonymity sets for different users therefore providing much less privacy than a mixnet. (See also issue #11 in ROBERT Github[11]).

Anonymity of upload authorizations. The ROBERT specification does not specify how uploads from Apps are authorized by the server. Such authorization is indeed essential to prevent false notifications.

The specific upload authorization mechanism, however, becomes important when Apps upload their observed EBIDs *separately* through a mix-net or from a NATted network connection. To ensure that the network-layer unlinkability is maintained, the *application*

[10] P. Richter et al. A Multi-perspective Analysis of Carrier-Grade NAT Deployment. ACM IMC 2016. https://www.icir.org/mallman/pubs/RWV+16/RWV+16.pdf
[11] GitHub Issue raised by Prof. George Danezis, University College London. https://github.com/ROBERT-proximity-tracing/documents/issues/11

level message, which includes the authorization, must be unlinkable as well. This is difficult to achieve using common cryptographic techniques and instead requires anonymous authentication mechanisms (such as anonymous credentials). Moreover, such anonymous mechanisms must be combined with a *rate-limiting mechanism* to ensure that these anonymous tokens cannot be abused to upload large numbers of observations.

Security Analysis

Key rotation

(See also issue #8 in ROBERT GitHub[12]) The protocols do not specify key rotation schedule, nor describe the consequences of key compromise and mitigations:

- Compromise of K_S implies that the one obtaining the key can decrypt **any** EBID and see all permanent identifiers.
- Compromise of K_G implies that the country codes are public, for **all** countries, as the key is shared.

Faking risk

As infected users upload their observed EBIDs, it is very easy to inject into this list the EBIDs of other users. An individual need only capture a target EBID, or potentially access it from a list of target EBIDs (e.g. of well-known people that have been intercepted and posted online), and adversary may relay the EBID of a target in such a way to ensure that the target is found at risk. There is no possible defense in this design, as veracity of encounters cannot be checked. In contrast, a decentralised system never uploads the EBIDs of non-infected individuals.

Risk of all data on a device being compromised

A risk that is not discussed in the protocol relates to its practical deployment. A practical deployment of this system requires constant transmission and recording of EBIDs at all times that an individual could be at risk of infection or infecting others. This requires a system to run in the background of a mobile device. An issue that applies in particular to devices manufactured by Apple is that background Bluetooth Low Energy usage of the type imagined in this document is forbidden. The new adjustments announced on April 10 to the specifications, in particular the introduction of a new Contact Tracing API, considering principles of data minimisation, only permit a protocol to query the operating system with a list of identifiers they wish to match or feed into an operating system–level risk scoring algorithm. They do not provide the functionality envisaged in ROBERT, which requires an infected user to upload the history of observed contacts. As a result, the only option for such applications is to use workarounds, notably to require a user to keep the app *foregrounded*, with the screen on and device unlocked and without passcode protection. **This means in practice, any user who has such a device stolen while in use or is asked to produce their device by law enforcement, has forfeited all data on the device**

[12] GitHub Issue raised by Prof. Serge Vaudenay, EPFL
https://github.com/ROBERT-proximity-tracing/documents/issues/8

not secured by further protection. According to data from the United Kingdom collected in 2016, about 446,000 people experience mobile phone theft each year, approximately 1% of all mobile phone owners.[13]

Conclusion

The above privacy and security issues relating to ROBERT are not theoretical in nature, and warrant serious attention and debate. We believe that adopting this system would open up significant avenues for systemic misuse and that it does not sufficiently prevent function creep or engender the trust that is crucial for public adoption, safety and legitimacy.

[13] Office for National Statistics, *Focus on property crime: year ending March 2016* (2016).