Security and privacy analysis of the document 'PEPP-PT: Data Protection and Information Security Architecture'

The DP-3T Project

19 April 2020

On 18 April 2020 PEPP-PT released a document on their GitHub repository illustrating the data protection and security architecture of the German implementation of PEPP-PT contact tracing (CT), which we will refer to here, as they do, as **NTK** for clarity.[1] In this document, the DP-3T project provides an analysis of this system to inform open discussion.

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 NTK centralizes the risk calculation at the server, rather than performing it locally on individual smartphones.

Overview of NTK

The system works in 4 phases:

1. **Installation**. When the app is installed on a smartphone, it registers with the backend server. The registration consists of a Proof of Work computation that returns a persistent identifier (PUID), which creates and associates the Android/iOS PUID that is necessary to push data to the phone.
   The PUID is used to derive ephemeral IDs (EBIDs) that are pushed to the phone to be broadcasted. EBIDs are generated by the server by encrypting the PUID with a global broadcast key that is renewed periodically, e.g. every 60 minutes. New EBIDs are pushed by the backend server periodically after installation. The global broadcast keys are deleted after 4 weeks.
2. **Normal operation**. Each app broadcasts the EBIDs that the server provided via Bluetooth and records the EBIDs seen in the broadcasts of other phones in close physical vicinity.
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*. Along with each EBID, the phone sends the time of contact, Bluetooth metadata (RSSI, TX/RX power), and optionally device information, and any other metadata required by the scoring algorithm run by the server.

[1] 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. The protocol is referred to as NTK diagramatically in the 'high-level overview' uploaded on 18.04.20.

4. **Contact tracing**. The backend server uses the global broadcast keys to decrypt the EBIDs, thereby recovering the PUID of the device(s) that were physically close to the diagnosed person and runs a computation to estimate infection risk. The server pushes a notification to the phones of people at risk, which it identifies using the PUID, to inform them of the situation.

Additionally, the collected data may be used for epidemiological studies (procedure not specified).

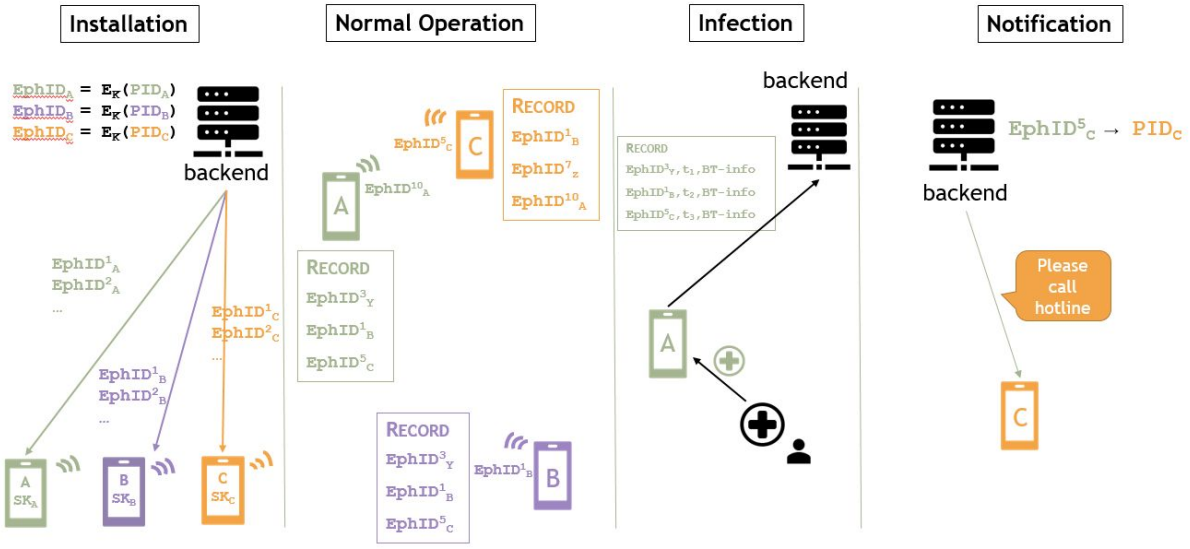

Figure 1. A description of the PTK system

Privacy analysis

This architecture raises the following privacy concerns. We also include concerns with both privacy and wider societal elements.

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 contact tracing tool, 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 -- with 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

pseudonyms to real identities and track individuals, infected or not, in their daily life by using the EBIDs generated from their PUID.

Second, given a target EBID (e.g., collected by law enforcement from a suspect, or at a passport control point), it is possible to **tag and classify individuals so that third parties can recognise them 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 *a specific, distinguishable* 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 a target group of people.

Learning co-location of users (social graph)

The centralized approach, by design, allows the backend to learn the entire contact graph of an infected individual. Moreover, because of the use of fine-grained epochs and timestamps, **the backend also learns encounters among non-infected users, in contravention of the need-to-know and data minimization principles expressed in the GDPR, as alternatives where this does not occur, such as DP-3T, are readily available**.

- **Contagion exacerbates learning of the social graph**. Exposure of this information is an even greater problem due to the characteristics of COVID19. Contacts of an infected individual are expected to become infected in a short period of time and the system must perform near-real-time, fine-grained tracking of physical interactions. Therefore, the backend learns not only isolated graphs of sick people, but more importantly, connected components that *quickly provide an accurate representation of social structures and users' interactions for a much wider population*.

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

- **Individuals in rich social graphs are easily re-identified**. We also note that rich graphs, such as the one that is created in the backend, are prone to de-anonymization attacks with little auxiliary information.[2] A recent study noted, for example, that knowing 1% of the network around individuals would mean observing 46% of all communications and quickly re-identifying a large fraction of the population.[3]

Learning infected close contacts

A tech-savvy user who is notified by the app that they have been in contact with an infected user can identify this individual. The procedure is as follows:

1. The tech-savvy user creates a number of accounts, one for every slot of the day in which he may want to identify an infector. For instance, to split the day in slots of 15 minutes, this user needs 96 accounts. NTK requires a proof of work and a CAPTCHA to register a new account. Since the NTK proof of work has to be simple enough to enable creation of individual accounts fairly quickly and CAPTCHAS solving can be outsourced at a negligible cost, we assume that 96 accounts can be created in a reasonable amount of time.

2. The tech-savvy user rotates the EBIDs of these accounts during the day, and maintains a record (e.g photos, a notebook) of which account was used in each slot, the EBIDs of users it observed, and the identities of the people he came into contact with during those slots.

3. When the user receives the notification that he is at risk in one of his accounts, this identifies the slot of the day in which the contact with the infected happened. From his records, the user can identify the infector.

Note that this attack does *not* depend on the information saved on the server, only of the local information of the adversary. Therefore it cannot be avoided. In fact, it is possible in any contact tracing system (including DP-3T).

Security analysis

There are two mechanisms in the centralized design that are prone to attack.

Faking Risk

As infected users upload their observed EBIDs, it is very easy to inject into this list the EBIDs of other users. An adversary need only capture a target EBID, or potentially access it

[2] See eg Ji et al. "Secgraph: A uniform and open-source evaluation system for graph data anonymization and de-anonymization." 24th USENIX Security Symposium (USENIX Security 15). 2015.
[3] Radaelli et al. "Quantifying surveillance in the networked age: Node-based intrusions and group privacy." arXiv preprint arXiv:1803.09007 (2018).

from a list of target EBIDs (e.g. of well-known people), and upload that EBID to ensure that the target is found at risk. There is no possible defense in this design, as the veracity of encounters cannot be checked. In contrast, a decentralised system never uploads the EBIDs of non-infected individuals and requires health-care authorization to upload the EDID of an infected person.

Risk of data compromise on unlocked devices

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 monitoring of the type imagined in this document is forbidden.

The new specifications announced on April 10 by Apple and Google introduce a new Contact Tracing API that, in accordance with the principles of data minimisation, only permit a protocol to query the operating system with a list of identifiers they wish to match or feed to an operating system–level risk scoring algorithm. They do not provide the functionality envisaged in NTK, which is necessary to obtain and upload the history of observed contacts. As a result, the only option for these 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 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 phone owners.[4]

Conclusion

The above privacy and security issues relating to PEPP-PT NTK 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 offer sufficient protection against function creep or engender trust in a manner that is crucial for public adoption, safety and legitimacy.

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