Best Practices

Operational Security for Proximity Tracing

DP-3T Team

Executive summary

In this document we describe security mechanisms that can be added to Proximity Tracing applications to ensure that the security and privacy properties provided by the protocols are not undermined by other components of the system. In particular we provide recommendations to protect:

- **Sensitive communications between app and server.** We provide guidelines to establish dummy traffic to ensure that network traffic that is associated with sensitive information (COVID-positive status, notified users) cannot be recognized by passive observers.

- **Communications between backend and smartphones.** We analyse the use of attestation as a support mechanism to protect the downstream communication. Attestation must be carefully considered as it increases the dependency on closed-source components and may not be available in all phones.

- **Metadata protection at the server.** We analyse what information could be collected by a server and provide recommendations in regards to logging and storage.

- **Validation of notifications.** We describe a simple mechanism to validate that when a user claims to have received a notification that the user is in possession of a phone for which the app has actually generated such a notification.

Dummy traffic: protecting against network adversaries

Adversaries that can observe network communication can observe network traffic between a user's smartphone and the backend server or servers. Powerful network adversaries might also be able to observe network traffic between backends. Network adversaries can use these observations to try to infer sensitive information about users: whether they received a positive COVID-19 diagnosis, or whether they received a notification of exposure to COVID-19 positive users.

We consider two types of network adversaries:

- Local network adversaries (e.g., a malicious WiFi hotspot).
- Major network operator (e.g., large telecom operators that can see both cellular and domestic internet traffic).

When apps use Google/Apple Exposure Notification protocol (or the DP-3T protocols), they need to communicate to the backend server, and potentially other servers, as determined by the health authority. When this network traffic is associated with sensitive information, it can reveal information to the adversary. Some examples of such sensitive network traffic are[1]:

- COVID-19 positive users' smartphones upload diagnosis keys to the backend server to enable proximity tracing.
- Smartphones of COVID-19 positive users obtain an upload authorization code from a health authority before uploading their keys to a backend server.
- Smartphones of users that have been exposed communicate this to the backend so that the backend can notify these exposed users (e.g., a system in which instead of asking exposed users to call an information number, it is preferred that a person calls those exposed users).
- Smartphones of users that have received an exposure notification provide proof of this notification to a server (e.g., to receive compensation or work leave).

These interactions originate from different **actions** performed by a user or the app. Actions can involve several network connections. For example, the first two are likely both caused by the action of *uploading diagnosis keys*. The third results from an action initiated by the app upon detecting exposure, and the last one corresponds to the action of *verifying notification*.

In this section, we propose a mechanism that provides users with **plausible deniability** with respect to these network interactions — i.e., a user can claim that an interaction is a

[1] At the time of writing, there is no implementation of interoperability between countries in which phones and/or backend servers exchange information. Depending on the implementation, some of these exchanges may also reveal sensitive information and may require the introduction of new mechanisms similar to those described in this document.

fake one generated automatically by the app. To achieve this, the main idea is to have all users regularly performing fake actions that, from the point of view of a network adversary, look like real actions. When observing an action, the network adversary cannot infer sensitive information about the user, as any action could be one automatically generated by the system.

Below, we provide guidelines for mechanisms that provide plausible deniability, including

how to create fake actions that are indistinguishable from real actions and how to evaluate the strength of the protection. For each of these elements, we provide example implementation of the techniques in the Swiss app SwissCovid.

Creating fake actions indistinguishable from real actions

This protection mechanism works by (1) producing fake actions that are indistinguishable from real actions and (2) distributing these fake actions over time. As a result, any observed action could, with reasonable probability, be a fake action.

Making fake actions look the same as real ones

Each action can cause multiple observable network events. For example, to perform the *action of uploading diagnosis keys* a smartphone in the Swiss system will:

1. Connect to the health authority backend to validate a Covidcode (a short validation code provided to COVID-19 positive users) and obtain an authorization token