Stopp-Corona App German TAN System

The PEPP-PT Consortium

20th of April 2020

Executive Summary

The Pan European Privacy-Preserving Proximity Tracing (PEPP-PT) is a framework for mobile applications tracing the proximity of users to Covid-19 positive persons and notifying persons which have been exposed to an extent that makes an infection likely. For this purpose, users who have been tested Covid-19 positive voluntarily provide their pseudonymous proximity history to have affected contacts notified automatically. A critical step in this process is the authorization of the upload for Covid-19 positive persons. One the one hand, the PEPP-PT system is built in a way that users remain pseudonymous all the time and it is thus not possible to assign Sars-Covid-19 test results to user accounts in the backend. On the other hand, reporting a user as Covid-19 positive will result in notification of (some of) their contact persons.Therefore, it must be avoided that attackers or even inconsiderate users incorrectly report themselves as Covid-19 positive.

The processes of testing, informing the users, and reporting positive tests to health authorities are specific to national and even regional legislation. Setting up a unified pan-European process might be desirable in the long term but is technically and politically infeasible to achieve in the time frame addressed by the PEPP-PT consortium.
This document describes two different processes that are in line with legislation and the processes of the German health system.it

Table of Contents

Introduction

The challenge we are facing is as follows: a pseudonymous user of a PEPP-PT-based mobile application has herself tested for Covid-19, receives a positive result and now wishes to report the infection via the PEPP-PT app to inform previous contacts which have been exposed to an epidemiologic relevant extent. This step must be impossible (or hard) for any user who has not received a positive test result and it must be easy for users with positive test results. Obviously, as the system never knows the real identity of the user, a naïve version where a backend server would receive names, birthdays, or any other personal data from a test laboratory and simply ask the user to authorize herself using this information is unacceptable. We rather introduce two flows based on Transaction Numbers (TAN) which slightly differ to factor in the different processes which are currently in place in Germany when health authorities (Gesundheitsämter) and/or test laboratories and General Practitioners are involved in the reporting.

To verify whether a person has been infected, we require that a TAN is communicated to or from the patient via an out-of-band channel (e.g. by phone or by mail). Such a TAN is necessary to ensure that only Covid-19 positive users can upload their proximity history (called CTD data) to the backend. TAN procedures heavily depend on the digitalization of the local health system and existing IT infrastructures. Consequently, the German TAN system is designed to integrate in a “brownfield” IT environment.

For Germany, two TAN approaches are defined at the time of writing. Both will be discussed in this document. Option 1 is the preferred proposal. However, due to overload of both health offices and clinics at the time of this writing, the introduction of a second technology as Option 2 is required.

Option 1: “Gesundheitsamt”-TAN

In this flow, an interaction between the Covid-19 positive user and the health authorities is required (e.g. by phone). The TAN is generated by the user, transmitted over an out-of-band channel to the authorities (e.g. read out by the user on the phone) and activated by the authorities in order to authorize a transaction (here uploading the proximity history to the server).

The notification process may only be triggered by authorized employees of the health organization. This may either be an employee of a local health office (“Gesundheitsamt”) or an employee of a dedicated call center of the center of disease control (“RKI Call Center”).

A login procedure must be established to access the backend and verify or guarantee that the logged in user actually holds this role. Unauthorized access to this functionality would result in an attacker being able to generate arbitrary "contact persons" himself and trigger the notification process. Such an attack would severely compromise the reliability and trustworthiness of the system and must therefore be prevented. We can establish the role of a health employee by relying on an external, sovereign identity provider. Alternatively, there must be a registration process for health employees on the backend, that determines the identity and authorization of such employees. This can be achieved by creating dedicated accounts for health organizations (Gesundheitsamt or RKI). A closed-network access to the respective backend interface is a less favorable option, as it would not impose an authorization of the actual user, but might be used as a migration path, facing the time restrictions for setting up novel identity management systems.

In this flow, it is possible to put harder restrictions on the lifetime of the TAN and a relatively short time of validity significantly reduces the time window for a successful attack on that TAN. This results in the possibility of keeping TANs short and easy to process for humans. However, it requires

both parties, the generator and the validator of the TAN to be online at the same time. The existing flow is as follows:

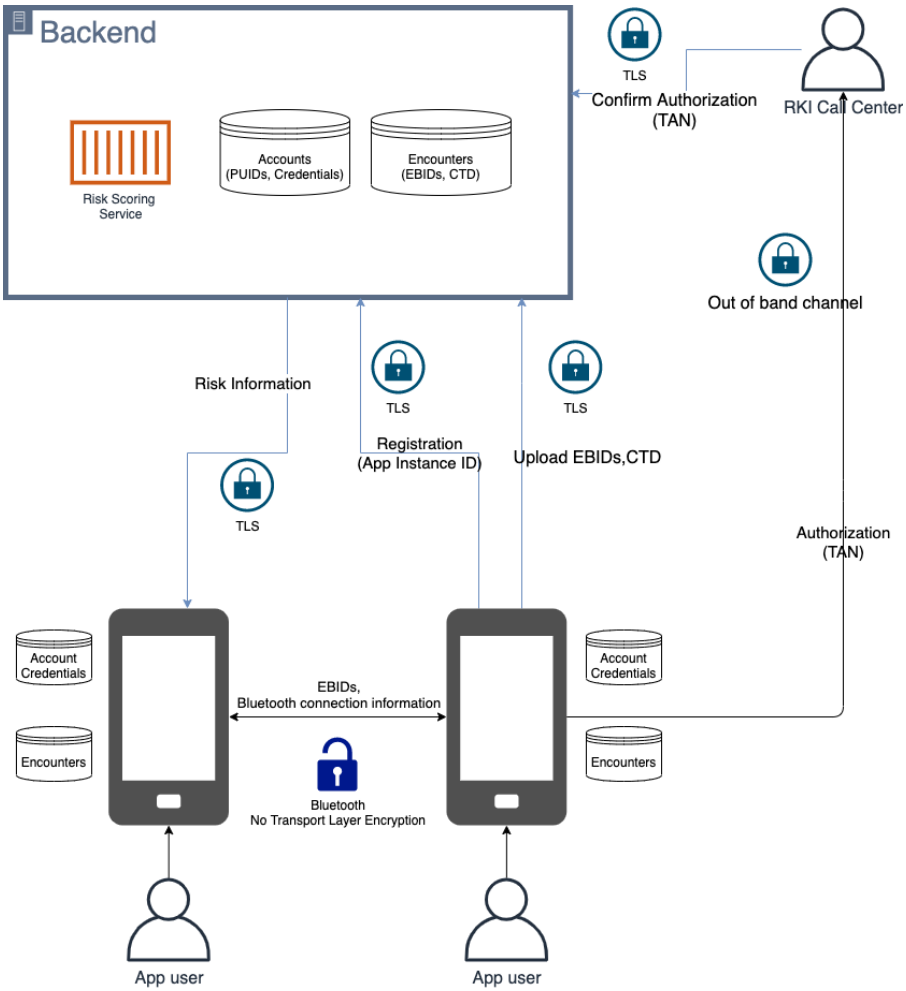
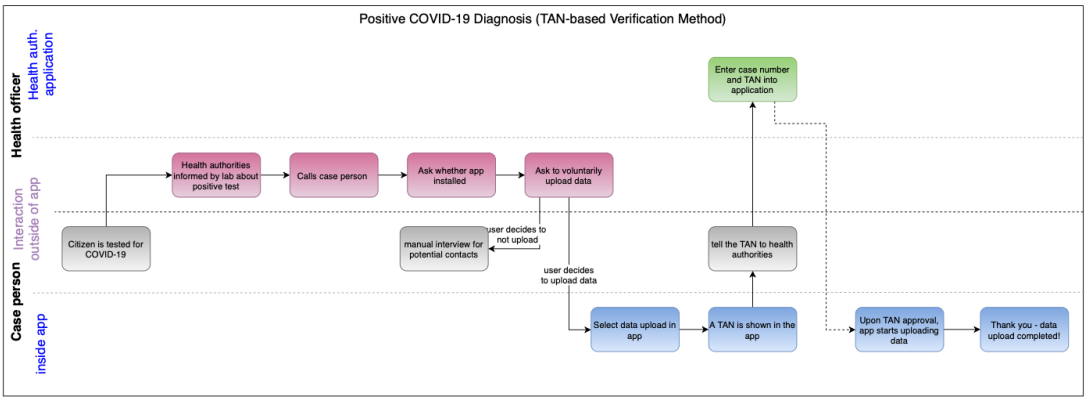


Figure: The architecture for the “Gesundheitsamt”-TAN. The “RKI Call Center” may optionally also be a “Gesundheitsamt”.

1. The user uses the app to generate a TAN on the backend.
2. Via an out-of-band channel, the user transmits the TAN to a health official.
3. The health official logs-in to the backend through a dedicated web interface.
4. The health official enters the TAN received from the user.
5. The app of the user is notified of the TAN activation and when the user gives final consent, the proximity history (CTD) is uploaded.



It is important that TANs are not assigned twice to different app instances. Duplicate TANs in the same validity time window would lead to incorrect notifications of users who are not or have not had anything to do with the actual case. Hence it is necessary to:

- Ensure sufficient entropy in TANs
- Limit the validity time of TANs
- Link a TAN to a specific PUID (App instance)

We propose that TANs consist of 8 randomly generated alphanumeric characters (0-9 and a-z). For 6 characters, the security level is $\log_2(36^6)$ ~= 21 bits. For 8 characters, the security level is $\log_2(36^8)$ ~= 28 bits. While this corresponds to a low security level, it has the advantage of reducing transmission errors and increasing usability. We also note that the TAN is transmitted verbally by phone and that massive brute force attacks are unlikely to be carried out over this channel. Further, in the case of such short TANs (8 characters), the validity time must be limited. A validity over a longer period is neither necessary nor conducive to confidentiality.

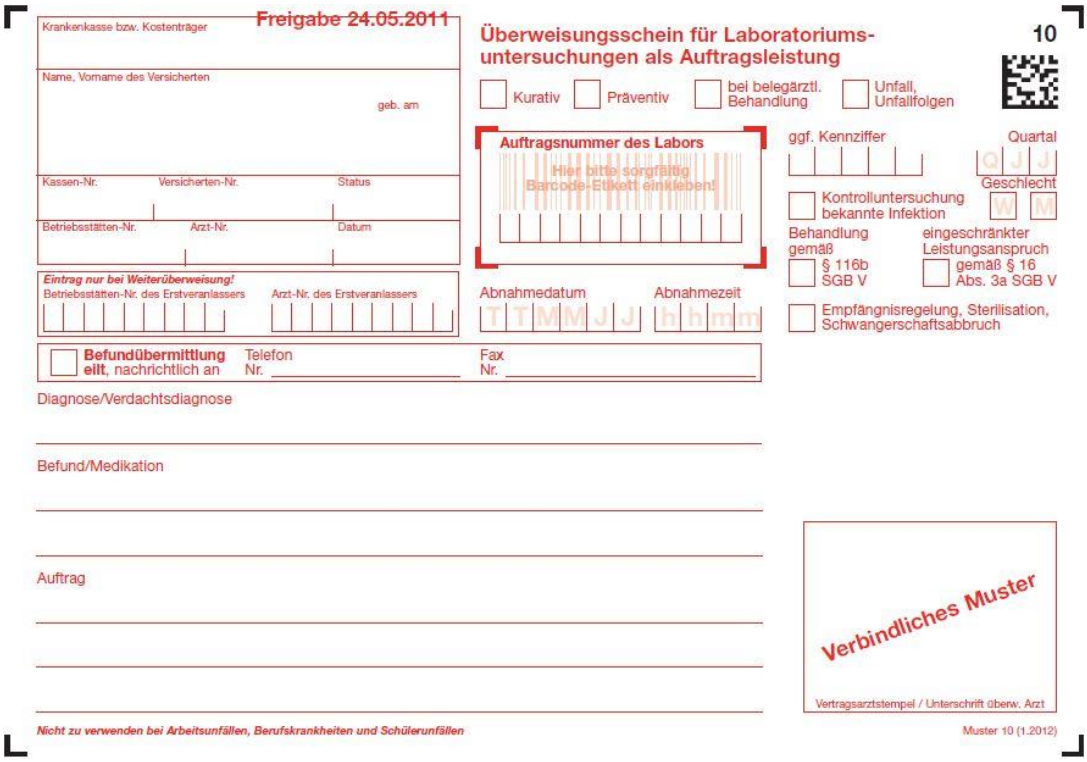
An example of a TAN as it would be displayed to the user is:

ZH56 98F7

Option 2: “Labor”-TAN

This flow is a “brown field” solution which integrates into existing text procedures in Germany where doctors/clinics use specialized laboratories to test for the disease. In the flow, we cannot rely on direct communication between Covid-19 positive users and health authorities. Instead, the backend connects to a "laboratory server" that is used to report pseudonymous test results from laboratories to the PEPP-PT backend.

In Germany, clinics, doctors, or test centers request Covid-19 tests from test laboratories using a so-called “Anforderungsschein” (see below), which is standardized for reimbursement purposes. The sample test tube is marked with a specific Order ID (OID) which is usually a barcode sticker. The laboratory is provided with patient data as well as the OID through the "Anforderungsschein". The patient receives the OID and the Lab ID as part of the visitation/checkup from the doctor/clinic.



Source: KBV, https://wiki.hl7.de/images/Kbv_muster_10.JPG

The Labor-TAN process is as follows:

1. The clinic/test center is provided a priori by a laboratory with sticker sets which have the OID as barcode printed on them. The doctor uses to stickers to: Label the “Anforderungsschein”, label the sample and provide the patient with the Laboratory ID and the OID on a flyer.
2. The patient is tested in a clinic and a sample is created.
3. The clinic fills in “Anforderungsschein” and provides the patient with the Laboratory ID and the OID.
4. The patient uses the app to scan the OID barcode or manually enters it into the app together with the Laboratory ID.
5. The laboratory receives the sample with OID and patient information from the clinic.
6. The laboratory provides test result to the backend for the given OID. The backend needs credentials to authenticate against the laboratory server.
7. The test result is communicated to the patient through the app and the upload is authorized. The backend therefore periodically checks for the test result every hour.

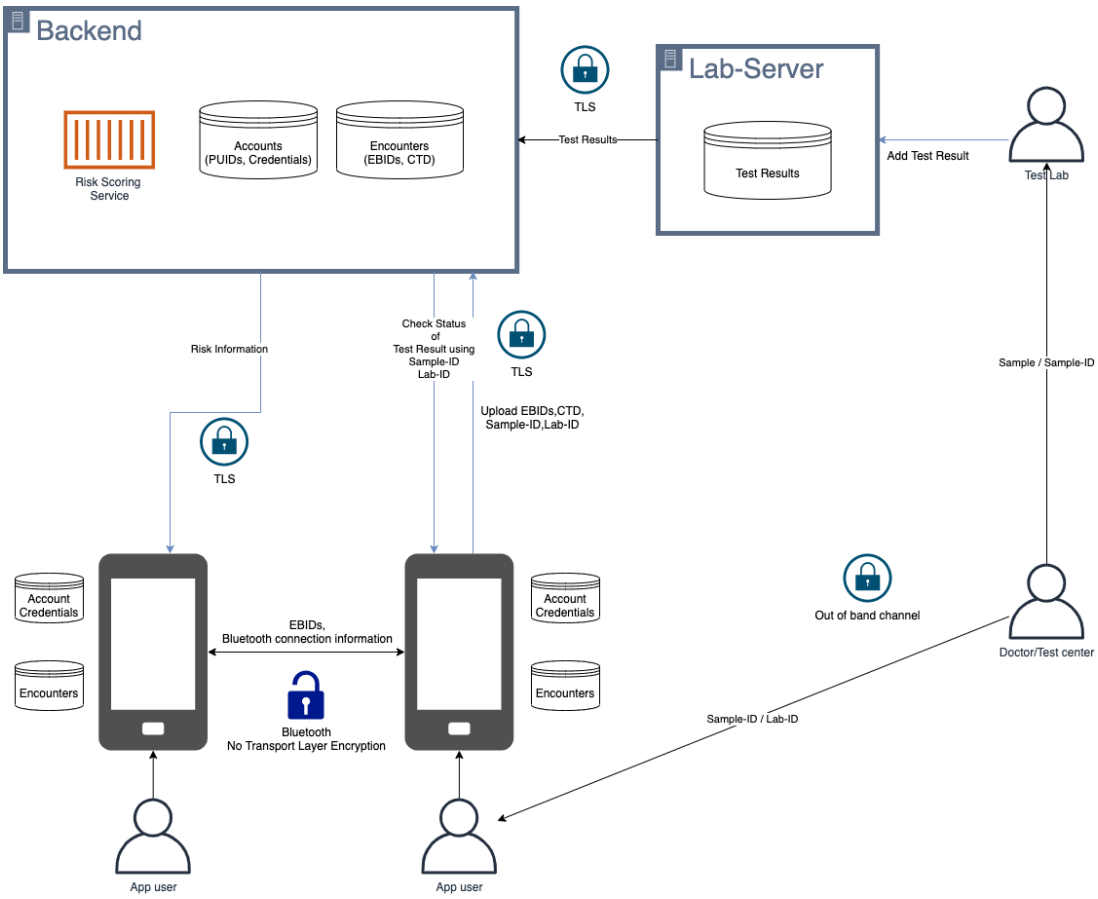


Figure: The architecture for the “Labor”-TAN.

The TAN in this flow is defined as follows:

Sample ID (“Probennummer”) := Laboratory ID | OID

- Laboratory ID: The unique identifier of the laboratory.
- OID: This number is unique to the test in the laboratory (“Auftragsnummer”). May be 8 digits up to 32 characters.

When a laboratory provides a test result to the backend, the Laboratory ID and OID are combined to the Sample ID (aka “Probennummer”). This SID is set active in the backend. **The laboratory server MUST authenticate the backend.** Backend authentication may happen through the use of TLS client authentication or symmetric credentials (e.g. API tokens).

In the app, a button is available which enables the user to upload data. When pressing the button, a dialog appears that informs the user and the user can give his consent to the data upload.

Upon consent, the user can now provide the Lab ID and scan the OID which has been handed out to him during the test. These are combined by the app to the TAN. The TAN is now sent to the backend which associates it with the PUID of the user.

Regularly, e.g. every hour, the Backend checks the result of the test by querying the laboratory server with the Sample ID. If the test is positive, the user may initiate the CTD upload to the backend.

While this flow can be used to integrate into the existing workflows of test centers and clinics, there remain residual risks. An attacker may try to create a significant number of accounts and squat OID numbers. The attacker may then also upload invalid/incorrect data. The current TAN flow is a compromise between security guarantees and practicability. To make "fake uploads" in this case more difficult the backend must ensure that:

- A Sample ID must be entered not earlier than 7 days before the result is available. This to some degree prevents pre-registration by “guessing” TANs once.
    - Note: Currently, positive tests account for 8% of results across Germany. The probability for an attacker to guess a SID that results in a upload is: The probability a guessed SID will have a result within 7 days times 8%.
- Different users may enter the same Sample ID. This prevents squatting of Sample IDs by an attacker by “guessing”.
- The backend only keeps Sample IDs associated with a PUID for 2 weeks. After 2 weeks, it is deleted and no longer checked. The upload must be initiated within the 2 weeks after the Sample ID is input.
- Registration of a SID which already has a result at this time is rejected.

NOTE: Due to the potentially short length of the OID (down to 8 digits), the system should in the medium term move towards deprecating any enumerable numbers. The health office should enforce the use of random OIDs by the laboratories of ideally 128 bits/32 characters (UUID format). The backend should then reject shorter OIDs. Due to the “brown field” character of this integration, we must expect some laboratories to still use enumerable, short OIDs.