PEPP-PT

Pan-European Privacy-Preserving Proximity Tracing

Data Protection and Information Security Architecture

Illustrated on German Implementation

Status: 20 April 2020

Executive Summary

Respiratory infections from SARS-CoV-2 are transmitted by droplets travelling over distances reaching 1.5 meters. A containment strategy, which is based on notifying individuals as early as possible that they have been exposed in an epidemiologically relevant extent to a confirmed Covid-19 positive person, is considered an effective measure to slow down the spread of the disease throughout all phases of the pandemic and has been recommended by epidemiologists all over the world. To date, this process has involved a manual reconstruction and notification of contact persons. That is, infected individuals must recall close contacts over the last 14 days and report them to health authorities. Health authorities must find ways to inform these contacts (e.g., research phone numbers or even addresses) that they have been exposed to an increased risk of a SARS-CoV-2 infection. This process is slow, requires considerable resources, and is error-prone.

This document describes a system to automatically trace and inform users in case they may have been exposed in an epidemiologically relevant extent to a confirmed Covid-19 positive person. The system complies with the PEPP-PT overall architecture and offers specific solutions for building blocks through the example of the German implementation. The real-world system must scale to millions of users within a few days and provide epidemiologists with measurement to analyze the spread of the disease. The system must preserve the user's privacy, take measures against de-anonymization of users—especially in relation to their health status—and, at the same time, be effective in containing the spread of the virus and provide interruption of transmission chains as early as possible. If the system were to leak information about personal behavior, identities, or even reveal who has been infected with SARS-CoV-2, users would quite rightfully refuse to adopt the system. If it were ineffective in reducing infections and the lifting of invasive lockdown measures, users would equally refuse adoption.

A description of a proposal for a joint French–German data protection and information security architecture can be found in [1].

Contents

Table of Figures

1 Requirements

We first introduce the functional and non-functional requirements of the PEPP-PT system.

1.1 Functional Requirements

F-REQ-1. Timely notification of contact persons

Obviously, the most important requirement is the ability of the system to notify relevant contact persons as soon as possible after a positive result from a Covid-19 test. The current process requires the infected person to provide contact details by hand and send them over to health authorities. The health authorities do further research and then contact the affected persons either by phone or by post. This contact-tracing process takes several days and does not scale to the dimension of tens of thousands of infected persons. As pointed out in [2], "manual contact-tracing procedures are not fast enough for SARS-CoV-2." However, they can be turned into an effective measure to limit the spread when supported by instant notification using mobile apps. The goal of supporting contact-tracing procedures with instant notification is to keep the delay between a positive test result and the notification as short as possible.

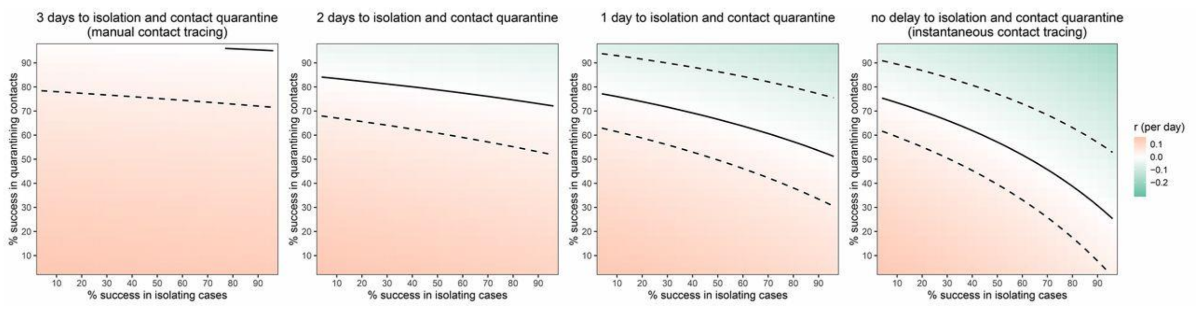

Figure 1 Quantifying intervention success, depending on delay to isolation (cf. [2])

F-REQ-2. Adaptive risk assessment

Receiving a notification does not necessarily imply that the contact person has been infected. Rather, each encounter must be evaluated by a "risk assessment" depending on factors such as duration and proximity of the contact, as well as overall epidemiological conditions. This risk assessment is the basis for the decision of whether (and how) a contact person should be informed. It weighs the potentially negative consequences of notifying users—such as causing anxiety and negative social and economical impacts (e.g., if the user self-quarantines)—against the necessity to inform highly exposed users and to detect infection paths. Ferretti et al. stress in [2] that this risk assessment must be done by a mechanism that can quickly adapt to the current situation of the epidemic. Such a mechanism is a trusted operational pandemic management backend. For instance, if infection rates become uncontrolled, a stricter risk assessment is required than when there are controlled transmission chains with few contacts. As the transmission properties of SARS-CoV-2 are not yet fully understood, an ongoing calibration of the risk scoring is required. The analysis required for adaptation will be conducted using the pandemic management framework in the pandemic management planning backend.

F-REQ-3. Epidemiological validation

Besides merely notifying contact persons, PEPP-PT must provide ways to assess its own effectiveness. This can help public health authorities to tailor prevention efforts. It can also help them to determine whether the infection risk scoring based on proximity and time duration needs to be adjusted. The epidemiological validation service should also statistically assess the accuracy of applied models. This validation is a necessary justification for whether the use of proximity tracing is in conformance with GDPR.

F-REQ-4. Support federation between backends of different countries
Users will encounter other users who are registered at a different backend. This will be the case in nationwide hosted backends. However, it may equally be the case in decentralized backend architectures where users are free to choose the preferred backend when registering the app. Though each backend is responsible for the risk scoring and notification of contact persons, it must be ensured that an encounter of two users with different backends is handled appropriately (i.e., that each backend notifies their users). This requires a federation architecture between different backends.

1.2 Non-Functional Requirements

1.2.1 Security Goals

F-REQ-5. Prevent Sybil attacks (attacker registering many accounts at the backend).

F-REQ-6. Temporary identifiers (EBID) received by the backend must be authentic.

F-REQ-7. Legitimate not-at-risk users of the system should not be falsely notified to be “at risk”.

F-REQ-8. Legitimate at-risk users of the system should be notified about being at risk.

F-REQ-9. Only legitimate users diagnosed with the infection should be able to upload their proximity history to the backend (authorization/authentication).

F-REQ-10. Only authorized personnel (i.e., backend administrators) should be able to access information stored in the backend.

F-REQ-11. Only authorized personnel (i.e., health authority officials and the backend administrator) can validate a TAN.

1.2.2 Privacy Goals

NF-REQ 8 The pseudonyms of participants (infected or not) should not be linkable to long-term identifiers.

NF-REQ 9 Infected users should be personally identifiable only by the health authority.

NF-REQ 10 PII of the at-risk people should not be learned by anyone.

NF-REQ 11 The location privacy of users should be preserved.

NF-REQ 12 Co-locations/partial social graph of not-infected users shall not be exposed to unauthorized parties.

NF-REQ 13 Users of the app should not be able to infer the partial social interaction graph of other users. An interaction graph reflects the social relationships of all users in the system: a labeled edge indicates an interaction between two adjacent users at a specific time.

1.2.3 Implementation Goals

NF-REQ 1 Scalability up to 100 million active users within a month
The solution proposed herein must potentially scale to a very large user base within a short time frame. That is, foreseeable performance bottlenecks must be considered in the initial design as there will be no time to gradually adopt protocols and develop new solutions to problems arising from a large number of users.

NF-REQ 2 Short-term availability of a stable release version at the end of the current lockdown phase.
At the time of this writing, the solution is expected to be release-ready within a very short time frame (i.e., a few weeks).

NF-REQ 3 Support all kinds of devices, especially low- to mid-range devices on Android and iOS.
Proximity measurements, in particular, are affected by environmental factors, hardware properties, and specifics of the communication stacks. The solution must work on the majority of devices, including outdated OS versions and cheap hardware components. For instance, to support 89.3 % of the Android OS versions currently used, the solution would have to support Android 5 (where 10 is the current version)[1].

NF-REQ 4 Integrate with brown-field IT systems and existing processes in the health care system.
At the time of this writing, very short time frames of implementation time are available to apply the solution. This makes academic approaches that would require stakeholders to fundamentally change their existing processes impossible. We have to acknowledge and consider that users will not be tech-savvy; that some health authorities may have limited IT possibilities; and that authorities, doctors, and laboratories may be able to slightly adapt their current processes but will not be able to introduce a new technology or make significant changes to their current mode of operation.

2 Threat Modeling

This section lists our assumptions, presents the actors and the assets, and lists the desired system properties (notably with respect to security and privacy).

1 https://developer.android.com/about/dashboards

2.1 Actors

The system has three different main actors. Later, we will introduce variants of these actors in the form of different adversaries.

2.1.1 Users of the Mobile Application

Anyone will be able to download the app from the Google and Apple app stores and install it on their phones—free of charge and advertisements. By users, we imply all users registered at the same backend. However, we acknowledge that, in a multi-national federation scenario, users might also have different backends.

2.1.2 Backend Administrators

Backend administrators run the infrastructures and the services of the backend architecture. They have access to log files and may start and stop services.

2.1.3 Health Authority Officials

Health authorities alone can authorize uploads of proximity traces by a Covid-19 positive user to the backend. They do so by following a process that is out of the scope of this document and involves a confirmed positive test result and direct contact to the tested person (e.g., by phone).

2.2 Assumptions

We treat the overall backend (including all of its subsystems) as a single entity. For instance, if the backend is honest-but-curious, then all of its subsystems are, and the adversary sees no communications whose source and destination is a subsystem of the backend.

We assume all credentials for authentication and access control are properly set and that only the intended party has access to said credentials.

We assume that all channels used in the system (e.g., those between backend subsystems and any health authority devices, or between the backend and the users) are integrity and confidentiality protected (e.g., with TLS with appropriate key pinning). However, our assumption is limited to what current protections TLS 1.2 provides (e.g., we include in our analysis that an eavesdropper can see packet destinations, timings, and volumes, which are not protected in TLS 1.2).

2.2.1 Adversarial models

We define five different possible adversaries:

In-Scope Adversaries

A1: Curious user (honest-but-curious)
A typical user of the system who will not look at any information not available via the App UI, nor will try to tamper with it. They behave normally and will not change their movement patterns in any way to learn more. The majority of users falls into this category.

A2: Tech-savvy user (malicious user)
This adversary includes black/white hat hackers, academic researchers, etc. This adversary has access to the system via the mobile app. They can set up BT, WiFi, and Mobile antennas to eavesdrop local traffic on all radio interfaces. They can decompile and modify the app to create a misbehaving client endpoint. They have access to the (open) source code of the app and the backend services. They will directly access backend services.

A3: Health Authority (honest-but-curious)
Health authorities have special privileges in that they have authority to grant uploads of proximity histories in accordance with the user. They are modeled as honest-but-curious actors who stick to the protocol. However, they may use information that is retrieved over the protocol to their advantage. Specifically, they can combine knowledge about infected individuals with proximity histories (and potentially public background knowledge) to learn more about infected, at-risk, and not-exposed individuals.

A4: Backend admins (honest-but-curious with privileged access)
Backend administrators can access all data stored at their servers and query data from the mobile apps within the content provider operational scope. They can also change the code of their backend software and the code of the mobile apps. We assume they will not modify the mobile app because doing so would be detectable. They can combine and correlate information, request information from apps, and supplement with other public information to learn (co-)location information of individuals.

A5: Eavesdropper (honest-but-curious eavesdropper who sees all network messages)
Different actors will be able to observe traffic either from the mobile phones or of the backend. This includes Internet Service Providers, local system administrators, Bluetooth sniffers, and exit nodes of mix networks (such as Tor). They can analyze traffic patterns, inspect payloads, and analyze sources and destinations of communication.

2.2.2 Out-of-Scope Adversaries

In addition to the aforementioned adversarial models, we acknowledge that there are further types of adversaries. However, we do not consider them to be realistic in the addressed scenario. In the following, we list these out-of-scope adversaries and explain why they are given this status.

A6: Malicious backend admins
Although A4 considers backend administrators to be honest-but-curious, we might also consider them to be malicious. In this case, they would not obey the protocols herein and could, for instance, leak secret information. At any time, a malicious backend administrator can delete the database, shut down API endpoints, and decrypt or craft new EBIDs. Depending on the backend architecture, they might also have access to secret BK_t keys. However, if an HSM is properly used, a malicious backend admin would not be able to extract the keys but could only use the HSM to their advantage. Fortunately, even the malicious backend admin does not have access to additional unpublic information that could be used to de-anonymize pseudonyms.

Nevertheless, we consider this attacker as out-of-scope because a maliciously behaving backend would be detected immediately, break several contracts and laws, and face high penalties and legal consequences.

A7: State-level adversary (rogue state)

State-level adversaries have the means to deploy large-scale technical attacks, may instruct law enforcement and intelligence agencies, and alter legislation. They have capabilities of A1, A2, and A5. In addition, they can obtain subpoenas that give them capabilities of the health authority (A3) or the backend (A4). They may want to obtain information about the population. They may also target particular individuals. They may be interested in past information (what is already stored) or future information (that will enable target tracing in the future).

This type of adversary is omnipotent and can run various attacks to de-anonymize users, fake data, etc. However, it is considered out-of-scope because it would require several legal entities to collaborate in an unlawful manner that would not remain unnoticed. The existence of a state-level adversary implies that existing social and legal norms are abrogated and civil stakes are at risk to an extent that goes far beyond what the adversary could achieve by exploiting PEPP-PT. It should also be noted that users can change their pseudonym at any time by re-installing the app and, thus, evade a continuous tracking by a state-level adversary.

In each case, we consider a polynomially bounded (computationally bounded) adversary (i.e., who is unable to break current cryptographic schemes).

3 Data Assets

We consider the following data that are stored on the different endpoints

- Stored on the phone
    - Set of current and future temporary pseudonyms (EBID) to transmit
    - Proximity history (CTD) of the last 21 days (containing the observed EBIDs and timestamps)
    - OAuth2 client secret for access to backend services (long term)
    - OAuth2 access token for access to backend services (short lived)
- Stored on the backend
    - Long-term pseudonym of an app (PUID)
    - OAuth2 client credentials of an app
    - Short term (1 h): access token (temporary client tokens)
    - Medium term (days to weeks): BK_t, EBIDs, CTD (containing EBID lists)
    - Push Notification Service ID (PID)
- Not stored: TAN

4 User Consent

PEPP-PT users will be asked to give their consent for every functionality of the app. The first aspect (proximity tracing) is the core functionality of the app and consent is required if the user wishes to start the app. Obviously, users are otherwise free to not use the app at all. All further functionality is optional and will require a separate informed decision by the users.

4.1 Consent to proximity tracing

After installation of the app and before registration at the backend, users will need to consent to

- the backend creating a random, unique, and persistent pseudonym, called PUID. Users will be able to change that pseudonym by re-installing the application at any time. In that case, the backend cannot link the new pseudonym to the previous one.
- the app using Bluetooth functionality to send out and receive Bluetooth Low Energy (BLE) advertisement frames. The advertisement frames will use temporary (ephemeral) pseudonyms, called EBID, to avoid tracking the user via BLE for a period longer than 1 hour.

4.2 Consent to Share Contact Information in Case of a Positive Test Result

In case of a positive Covid-19 test result, the user will be asked to give consent for sharing of the previously recorded history of temporary pseudonyms of nearby devices (including respective timestamps) and sharing information with the affected contact persons that they have been exposed to an increased infection risk. No information about the location or exact time of the contact will be provided.

4.3 Consent to Data Use for Research Purposes

Independent from the notification of contact persons, information about epidemiological validation is valuable epidemiological data that helps to understand the spread of the disease. Users may opt-in into sharing pseudonymous data for epidemiological research purposes.

4.4 Consent to App Telemetry

App providers may need to monitor adoption rate of the apps, especially the number of active users or crashes on exotic hardware platforms. Users may give their consent to send anonymous telemetry data (OS, version, and hardware info) to the app provider's backend.

5 System Design

This chapter provides an overview of the PEPP-PT system and introduces the protocols for the individual steps in the user journey.

5.1 Architecture Overview

The system consists of three major components:

- An app is installed on the mobile phones of the users and ephemeral pseudonyms transmitted via Bluetooth (EBIDs) are collected and stored. It also broadcasts an own EBID.
- A backend is responsible for generating the EBIDs that are broadcast by users. The backend also processes EBIDs that have been observed and uploaded by infected users within the last 21 days and runs a risk assessment to determine the contacts that are epidemiological relevant (i.e.,

exposed to a degree that makes an infection very likely). Note that in case of multi-national federation, this backend is country-specific.
- Notifications could be regularly pulled from the backend by the app. However, due to technical limitations that are discussed in detail below, a push notification service is used to trigger contact persons and have their app pull the notification message from the backend.

The following figure illustrates the components and the data they store, as well as the interactions between components and users.

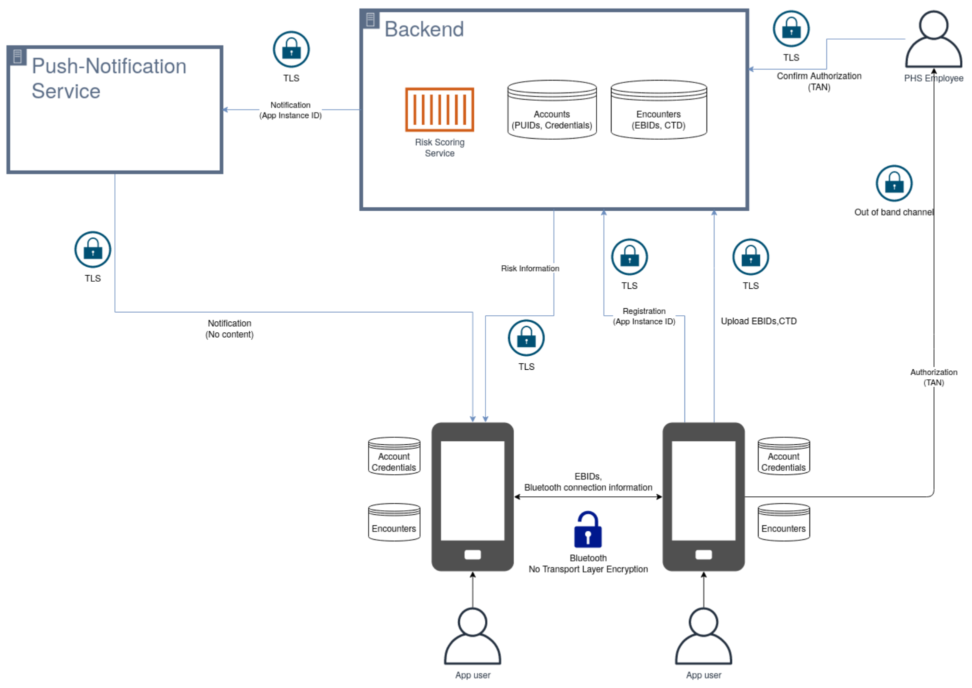
Push-Notification Service
Backend
Risk Scoring Service
Accounts (PUIDs, Credentials)
Encounters (EBIDs, CTD)
TLS
Notification (App Instance ID)
TLS
Confirm Authorization (TAN)
PHS Employee
Out of band channel
TLS
Risk Information
TLS
Registration (App Instance ID)
TLS
Upload EBIDs,CTD
Notification (No content)
TLS
Authorization (TAN)
Account Credentials
Encounters
EBIDs, Bluetooth connection information
Bluetooth No Transport Layer Encryption
Account Credentials
Encounters
App user
App user

Figure 2 High-Level Overview

We assume that all communication between backend and app is secured via TLS and that the server is always authenticated via appropriate certificate pinning (TLS server authentication). Upon successful (anonymous) registration, the app is authenticated by the server using account credentials or access tokens (OAuth2.0).

The communication between the backend and push notification service is also secured via TLS. No payload data are transferred via this channel; it only serves as a signaling mechanism. The backend authenticates against the push notification service using service-specific credentials.

Public health service (PHS) employees or laboratories communicate with the backend via TLS. They authenticate on this channel using account credentials (OAuth2.0) and are able to authorize contact and time data uploads.

In order to authorize app users to upload, TAN information is exchanged between user and PHS entities through an out-of-band channel. The details of this TAN transmission depend on national specifics (e.g., organization of health ministries, public health offices, laboratories, and doctors) and must consider possible restrictions (e.g., personnel capacities, technical limitations, lack of technical expertise at the users' side, etc.). In Germany, a "TAN flow" enables authorization by public health offices and a barcode-based "Laboratory flow" enables users to immediate access the test results provided by test laboratories. If tested positive, this gives them the choice to upload their contact history. The details of these flows are out of the scope of this document.

Apps receive and send ephemeral Bluetooth pseudonyms (EBIDs) via broadcast. As this is an open broadcast channel, it is not secured.

5.2 Protocols

In the following, details are given on the protocols between the components of the architecture, structured by the typical user journey. This journey consists of the following steps:

- The user installs the app, gives consent to basic proximity-tracing, and registers at the backend.
- During normal operation, the app is in proximity-tracing mode.
- In case of a confirmed Covid-19 diagnosis, the user may consent to sharing the proximity history with the backend and informing relevant contact persons.
- Contact federation is a process unnoticed by the user that supports notification of contact persons who are not registered at the same backend.

5.2.1 User Registration

To address requirement F-REQ-5, attackers must be prevented from mass creation of user accounts. The traditional way to avoid such Sybil attacks is to require user authentication based on identity attributes (e.g., email accounts or phone numbers) that cannot be created infinitely. As the design of PEPP-PT avoids the collection of any personal information, we use a different approach to limit mass registrations by malicious users. This involves the combination of a proof-of-work (PoW) with a Captcha; the former makes mass registrations expensive and prevents DoS attacks by qualified-yet-unauthenticated requests and the latter requires human interaction. After the registration protocol, the backend possesses a 128-bit unique random pseudonym of the user, called PUID. The app will possess OAuth2 client credentials [3] , which will be used to authenticate all further requests to the backend. Client credentials consists of a *client_id* and a *client_secret*—both generated randomly—transmitted over TLS to the app after successful registration and stored securely in the phone's keystore. Following an OAuth2 client credential flow, the app will need to authenticate itself towards the backend for all further API requests by retrieving an *access token* in exchange for the client credentials. The access token is an OAuth2 bearer token in the form of a JSON Web Token [4] that is valid for a limited time span and must be refreshed using the OAuth2 refresh flow when expired.

The registration protocol between the app (and its user), the authorization API, and the main backend is as follows:

App creates Push Notification ID PID

App → Push Notification Service:	*PID*
App → Registration Service:	Request to register, *PID*
Registration Service → App:	*Cp* with difficulty *D*, *Ca*

	User solves captcha(Ca)
	App computes pow(Cp)
	Registration Service creates random PUID
App → Registration Service:	Solution to *Ca*
	Registration Service verifies solution Ca
App → Registration Service:	Solution to *Cp*
	Registration Service verifies solution Cp
Registration Service → *App:*	*cred*
Registration Service → *Backend:*	*PUID, cred, PID*

App stores:	*cred*
Registration Service stores:	–
Backend stores:	*PUID, cred, PID*

Where:

Cp:	64-bit random proof-of-work challenge
D:	Difficulty of the PoW challenge - a pre-configured integer between 0 and 256
Ca:	Captcha challenge
PID:	Push Notification ID
PUID:	128-bit secure random number
cred:	OAuth2 client credentials as defined in [3], consisting of a random client ID and a client secret for use in the "client credentials" authorization flow
pow(•):	A proof-of-work algorithm. We propose pow(•) based on scrypt, as defined in RFC7914 [5], or Argon2, defined in [6]. A less mature alternative might be nano PoW [7]

The purpose of the proof of work is to make mass creation of accounts unattractive—especially in combination with a strong Captcha. If scrypt is used, it should be noted that scrypt has not been designed as a PoW algorithm. Therefore, using it as such requires a choice of parameters that is not in line with the original RFC7914 [5], which proposes scrypt as a key derivation function. However, scrypt PoWs are used by various blockchain systems such as Litecoin, Dodgecoin, and Feathercoin, and have shown in the past to surpass SHA-256-based PoW (as used by Bitcoin) because they are memory bound, rather than CPU bound and, thus, do not allow significant advantages to ASICs. When choosing scrypt, the following parameters may be a reasonable choice:

- Our input *P* to the scrypt algorithm is defined as the concatenation of a 64-bit nonce with the 64-bit challenge, that is, *P* := *nonce* | *challenge*
- The salt is the static value "PEPP-PT-POW"
- *N*=2 is the so-called cost factor
- *r=8* is the block size
- *p=1* is the parallelization parameter

N and *r* define the memory consumption for the computation of the hash, which is 128•*N*•*r* bytes.

The app must try to find an input *nonce* to solve the proof of work and the user must solve the Captcha. Once the backend has received and verified both solutions, the *PUID* is marked as active. If within a certain time span (e.g., a few days), the backend has not received both solutions, the PUID is deleted.

If the app provides a wrong result, the temporary account is purged to prevent abuse of the endpoint to calculate the PoW remotely. As long as the app is not fully registered, upload and notification functionality will not be available.

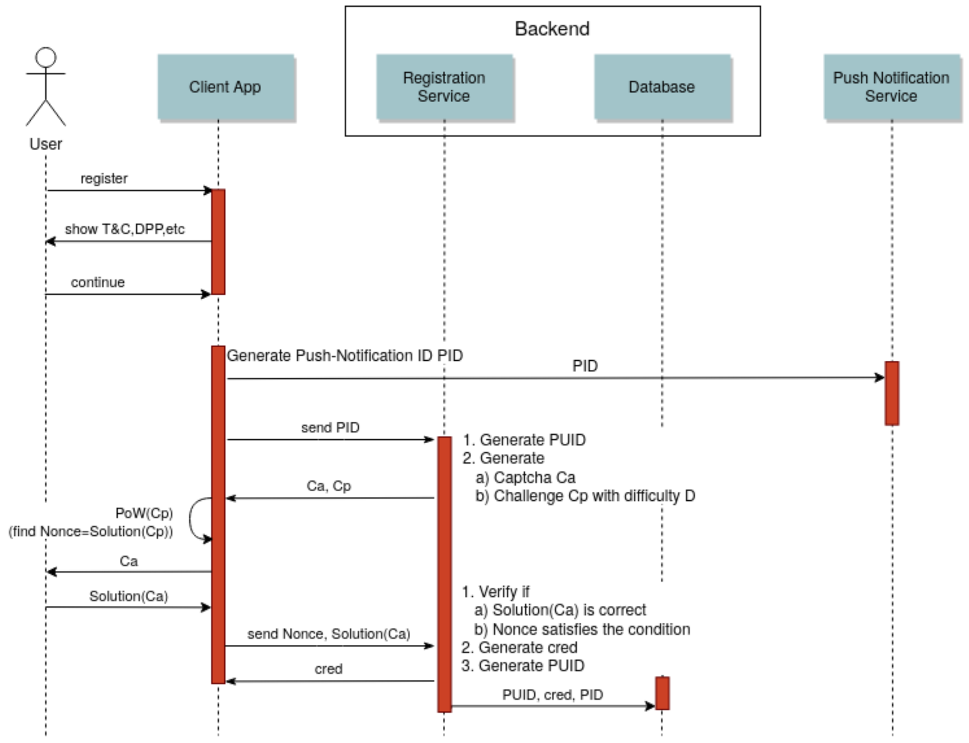

Figure 3 Registration Flow

After registration, whenever the app communicates with a backend endpoint, it uses its OAuth2 client credentials to retrieve an access token. The access token is only valid for a limited time. OAuth2 is the standard protocol for token-based authorization, which allows us to use short-lived credentials for authorization at backend services when needed, and only requires long-term credentials when requesting tokens from the Identity and Access Management (IAM) system of the backend.

5.2.2 Proximity Tracing

Note
The following section describes the current German implementation of PEPP-PT. For the corresponding French version of creating temporary pseudonyms, we refer to the description of ROBERT [1]. Both approaches are currently federation compatible and will likely be unified in the near future.

After registration, the backend regularly generates global secret keys BK_t, which apply to all users and are valid for a short timeframe t (e.g., 1 h), as well as Ephemeral Bluetooth IDs (EBID) for all users by encrypting their PUID:

$EBID_t(PUID) = \mathrm{AES}(BK_t, PUID)$

On request of the app, the backend generates enough EBIDs for the app for a timespan in the future (e.g., 2 days). The app stores the EBIDs to ensure that it will not run out of valid EBIDs if the phone does not have an Internet connection for a while.

Thus, the protocol for the key retrieval is as follows:

App	→	Backend:	*cred*
			Backend verifies that cred are valid client credentials
Backend	→	App:	*token*
App	→	Backend:	Request to retrieve broadcast IDs, *token*
			Backend retrieves PUID of token
			Backend calculates $EBID_t$, EBID_{t+1}, EBID_{t+2}, …, $EBID_{t+k}$ *as a function of PUID (see above)*
Backend	→	App:	$EBID_t$, EBID_{t+1}, EBID_{t+2}, …, EBID_{t+k}

App stores: $EBID_t$, $EBID_{t+1}$, $EBID_{t+2}$, …, $EBID_{t+k}$
Backend stores: BK_t

Where:

cred:	OAuth2 client credentials as a result of the registration flow
token:	OAuth2 JWT bearer token, including the client ID
PUID:	Permanent pseudonym of the app
$EBID_t$:	Ephemeral pseudonym for the app for time slot t

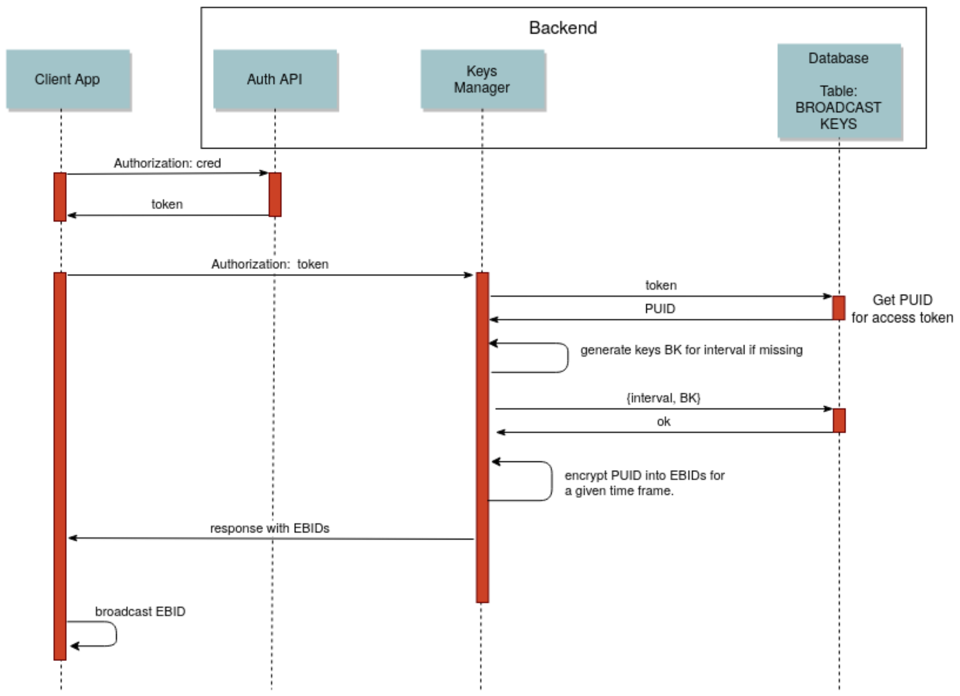

Figure 4 Retrieving EBIDs from an app

At this point, the app begins to broadcast the currently valid $EBID_t$ via Bluetooth Low Energy (BLE). The backend must be able to determine the PUID associated with a given $EBID_t$. While the backend must be able to determine the original PUID, other participants—in particular other mobile apps—must not be able to make this determination. The $EBID_t$ is periodically renewed to prevent location tracking of users and is broadcast via BLE advertisements using the Bluetooth Low Energy privacy feature. This feature is available as of Bluetooth 4.0 and uses regularly changing Resolvable Private Addresses (RPK) instead of fixed hardware addresses to prevent tracking of users who send out continuous BLE advertisements [8]. The implementation must ensure that whenever a new EBID is used, the RPK is changed as well to avoid linking of these two identifiers[2].

The app is constantly scanning for other PEPP-PT apps and their Bluetooth broadcasts by subscribing to the Bluetooth Service ID of the PEPP-PT app. The app records all EBIDs received by the scan together with the current time and metadata of the Bluetooth connection and, optionally, device information and device state. The metadata are used for the risk scoring algorithm and include non-personal data that allow a more detailed interpretation of proximity history. This includes RSSI, TX/RX power and—optionally after user opt-in—further data such as WiFi state and display state that help to calculate more precise distance measurements. It does not contain any personal identifiers and it does not contain the location of the user. We refer to these data as Contact/Time data (CTD). These data will initially only be held and processed on the mobile device until a user is confirmed as infected. In case of infection, the CTD may be used to estimate the duration of a contact and the distance between the users and, thus, determine a risk level. The CTD will be deleted from the phone after the epidemiological relevant time (e.g., 21 days) has passed since the contact occurred.

[2] In Android, the RPK is at least changed when calling BluetoothLeAdvertiser.startAdvertising()

5.2.3 Infection Notifications

When a user is classified (out of band) as Covid-19 positive, the CTD (including the timestamps and EBIDs of individuals who were in proximity to the patient) must be provided to the backend to initiate a process (in the backend) that runs the risk assessment to evaluate those contacts who might be at risk and to notify these "at-risk" contacts. Referring to F-REQ-9, this step is crucial and requires a strong authentication of the user that is Covid-19 positive. Otherwise, attackers may pass themselves as patients and corrupt the contact data and process in the backend.

The contact notification process in the backend is as follows:

- The user should be authenticated with an out-of-band means and notified about the positive test result. During this process, the user obtains a TAN that is accepted in the backend (e.g., the TAN could be transferred by phone). The process of distributing and activating the TAN, however, is not part of this document as it must fit the organizational structure of the respective national health care system and is tightly coupled to the process of informing the user about a positive test result, which typically involves the patient's general practitioner and the health authorities. A discussion of two possibilities to implement a TAN system for the German infrastructure is carried out in the document paper-based TAN system as described in [9].
- The app uses a REST endpoint of the backends and uploads the recorded CTD data. The upload is authorized by a valid TAN. The backend holds the CTD for up to 3 weeks.
- Using the timestamp t of the entries in the CTD, the backend is able to determine which BK_t was used to generate the EBIDs in the CTD entry. Then, the backend decrypts the EBID to the corresponding *PUID'* with $PUID' = AES^{-1}(BK_t, EBID)$.
- Finally, the backend updates the risk of the *PUID'*. The risk calculation algorithm is out of the scope of this document and will consider epidemiological factors such as closeness, duration of contact, and variance of signal strengths. The purpose of risk scoring is to filter out all contacts who were in proximity of the user, but were not exposed to a relevant degree, as they do not need to be informed.
- For all affected *PUID'* and a large number of randomly selected other users, a push notification is sent to the app. This message contains only a random number or the hash of an actual message. It only serves to "wake up" and trigger the app to send a request to the backend asking for its risk. If the random number/hash in the push notification message matches a message in the backend, the backend will send the respective content to the app, inform the user about a potential exposure to a Covid-19 positive person, and provide instructions on how to proceed. By having standardized message formats which make "real" messages indistinguishable from "noise" messages, as well as a significantly larger set of randomized receivers than actual addressees, it is not possible for an eavesdropper [neither a push notification service like Google Firebase Cloud Messaging (FCM) or Apple Push Notifications (APN), nor the Internet Service Provider (ISP)] to determine whether the target of the communication has a risk of being infected or not.

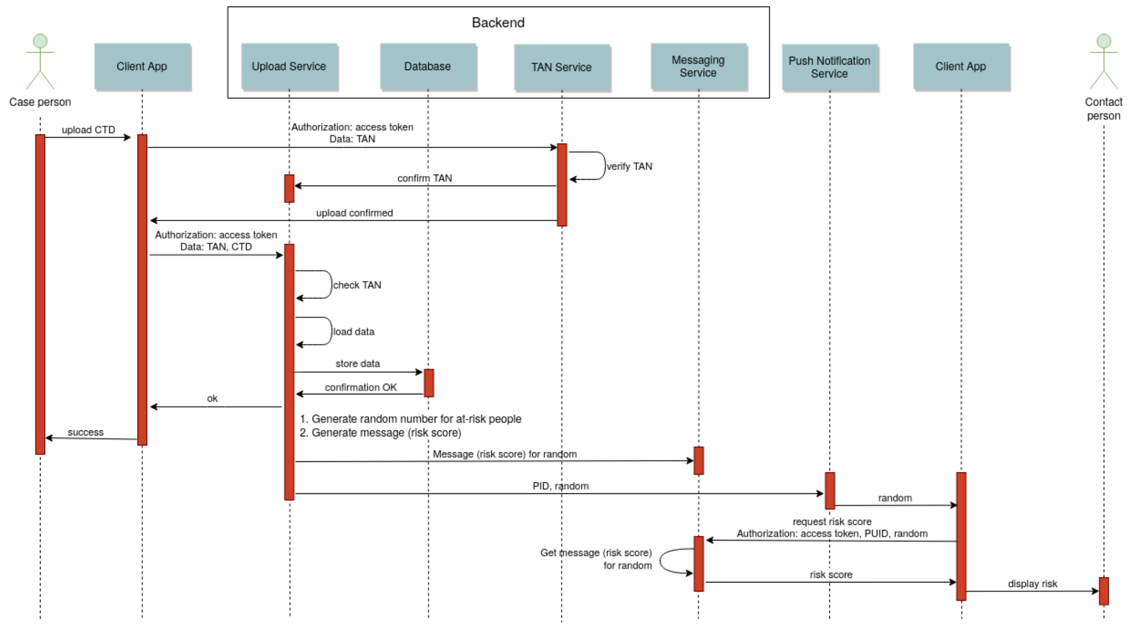

Figure 5 Uploading CTD and informing a user at risk

The TAN must have been exchanged out of band between the backend and the case person before this flow. In this example, one of the German TAN flows is used.

Note: The calculation of the risk score in the backend and not on the phones of individual users is required for three reasons:

1. *Functional*: It is a functional requirement (F-REQ-2) that the risk assessment algorithm can be quickly adapted to the current developments of the SARS-CoV-2 epidemic. That is, it must be possible to immediately react to uncontrolled infections by increasing sensitivity of the risk assessment (or reduce it, accordingly), as well as to factor in future insights into the transmission properties of the virus. If the risk assessment algorithm resides in the app, the only way change it would be through the mobile clients. This process would lead to a fragmented system of various clients running different versions of the risk assessment, rendering the main purpose of the app—(F-REQ-1) and (F-REQ-3)—void.
2. *Security*: The risk assessment seeks to maximize true positive notifications and to minimize false positive notifications according to the national public health policy. A false positive notification would result in a user who is at a very low risk of infection to be informed. This could have a psychological and economical impact on the user (e.g., if the user self-quarantines) and may even impact national economics, if occurring on a large scale. False positives may be caused by incorrect or spoofed proximity data. They can also result from insufficient authorization of Covid-19 positive users who upload their proximity history. This process must be tailored to the possibilities and conditions provided by the public health care system. If several authorization processes are supported, the central risk assessment can factor in that some Covid-19 reports are less reliable than others.
3. *Privacy*: The risk assessment algorithm ensures that only users who are actually at risk will be notified. If the risk assessment is decentralized to the clients, all users would need to be informed every time about all potential contacts—even irrelevant ones—and local risk assessments would then discard this information in most cases. This is a violation of the "data minimization" principle,

stated by Article 5 (1) (c) of the GDPR: "personal data shall be […] limited to what is necessary in relation to the purpose".

5.2.4 Federation across Europe

Note
A detailed description of the federation concepts is out of the scope of this document. As this concept is aligned with those of other countries (esp. [1]*), minor aspects such as the EBID generation will change.*

Another requirement of the system is to facilitate the operation of multiple backend services (cf. F-REQ-4). Especially in Europe—where the freedom of movement is an important aspect of daily life—it can be expected that international contacts are common.

At the same time, the system should allow countries to exercise data sovereignty of its own solution and service. Each country should be able to run its own backend service and must be able to decide:

- How EBIDs are constructed
- How risk analysis is done
- How and by whom the backend services are operated

Each domain (e.g., country) operates its own app and backend including the key-server generating and storing the BK_t keys. Each domain uses its local system and its own set of keys for mapping EBIDs to PUIDs. Note that each domain only has access to its own backend.

To achieve this, it is necessary for a backend system to determine from which backend system any EBID originated. This is ensured through the use of an encrypted country code (ECC) encoded into every EBID. Backends that did not issue a given EBID are unable to translate it back to the respective PUID. Risk analysis is always executed by the "home" backend of the user subject to the given EBID. Only the "home" backend decides if the user must be notified ("needs to know"). The origin of the forwarded EBID can be used as part of the risk analysis.

Such a design allows users (resp. app) to collect EBIDs issued by any backend and the backends to process and potentially forward them.
In order to achieve those properties, a federation between backends is defined including the following standardized interfaces:

- Backend-to-backend interfaces used to exchange "contacts at risk" (EBIDs)
- Routing information encoded in all EBIDs

An EBID contains a prefix that uniquely identifies the domain of origin. The prefix (ECC) 1 byte, for example:

	ECC (1 byte)	Domain-specific ID (15 bytes)
EBID	[0x00-0xFF]	123456-789a-9b9c-9d9e-9fa0a1a2a2a3

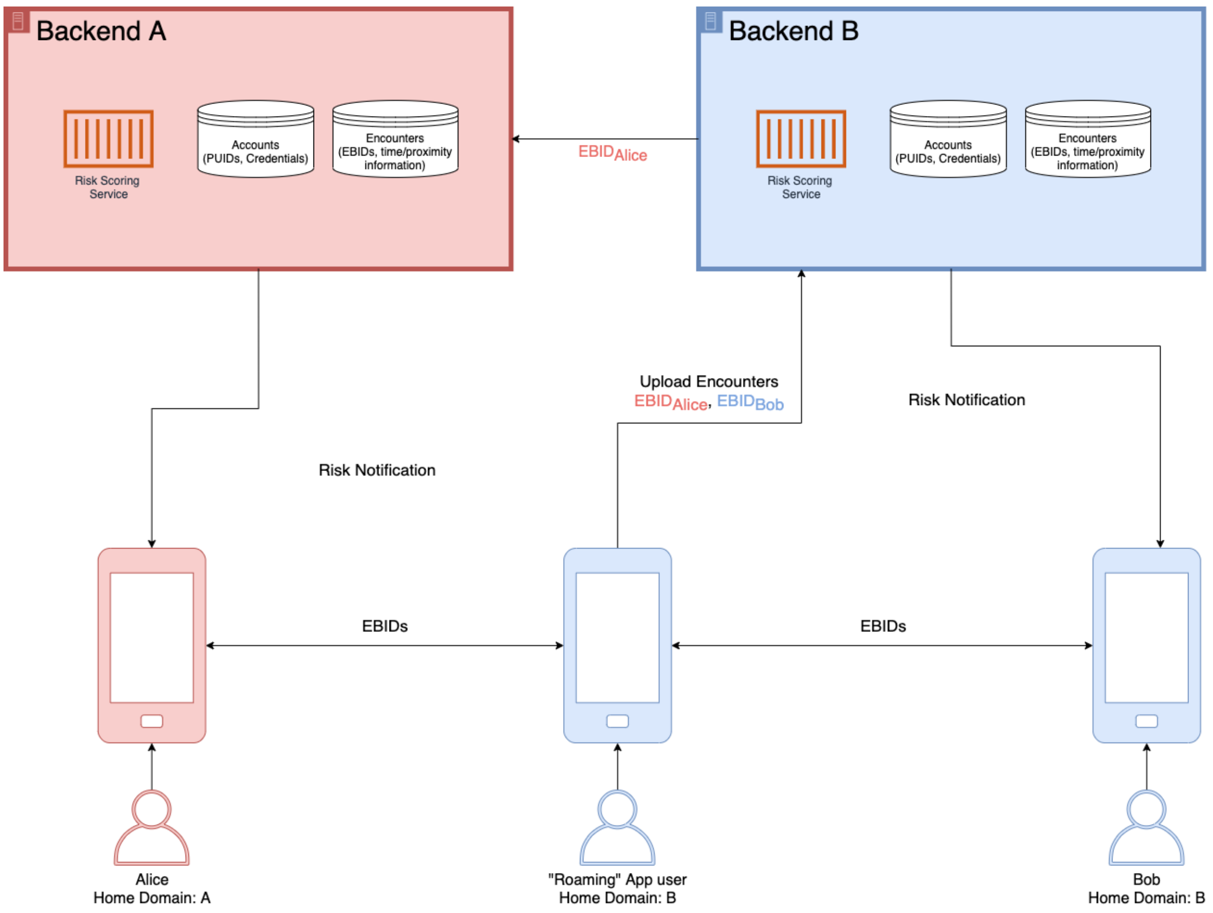

Figure 6 High-level roaming architecture

5.3 On Push Notifications

Note
The following section describes the current German implementation of the PEPP-PT. The corresponding French version ROBERT [1] strives for a pure polling-based approach to avoid the usage of push notification services.

To notify the app about an event affecting its user, third-party push notification services such as Firebase Cloud Messaging (FCM) and Apple Push Notification Service (APN) are required. If not used properly, this service can infer information about users being infected or at-risk either by the contents of a message or just the presence of messages and traffic. To prevent the operators of the push notification services from gaining such insights in users' health status, the messages sent via this service must not leak any such information. In this section, we describe why using such a service is necessary for the operation of a system that should scale to millions of users and present how we use the push notification services and overcome the concerns coming with the use of such services.

All protocols presented above are built such that they are operable without the use of a push notification service. However, the app needs to query the API exposed by the backend either in regular time intervals or when the app is explicitly triggered to do so. This introduces several technical challenges to the app and the backend that can be addressed by using push notifications:

1. If a huge number of apps regularly queries information from the backend, this adds (mostly unnecessary) workload to the backend. For instance, a main purpose of the app is to tell users if they are at risk of infection from previous contact or (eventually) to tell them about their test result. To enable a quick reaction of the user, this notification should happen with the

shortest possible delay (i.e., within a few minutes rather than hours or days) as shown by epidemiological studies [2]. For a naïve polling approach, with an estimated user base of 50 million users in Germany, this would mean that the backend has to serve 50 million requests per hour, assuming that a user should receive the test result or at-risk status with a maximum delay of one hour. This would result in 13,888 requests per second for the messaging alone, including establishing a TLS connection and doing a database lookup, assuming the best case of an equal distribution of requests over time. This is well beyond a reasonable architecture that could be operated by any non-hyperscaler (such as Amazon, Google, or Microsoft) and set up in the given time frame.

2. Apps are frequently suspended by mobile operating systems to save resources and limit their ability to perform potentially harmful actions. It may not be possible to bypass these mechanisms without having special rights on the phone.
3. If an app does not poll the results from the backend for a long time, the data have to be stored until the client finally sends the request. Thus, sensitive data are available for a longer amount of time in the backend.

Sending push notifications can overcome these issues by triggering only a relevant fraction of all apps to request results in a timely manner and, therefore, reduce unnecessary workload on the backend while providing any important information to the app as soon as it is available. Being deeply integrated in the mobile OS, the services can also operate when the app itself is not running and, thus, "wake up" the app so that it can execute a routine.

While it is possible to directly send a customized message over the push notification service, this would enable its operators to retrieve the information about the user's health status (e.g., infected, not infected, or at risk of infection). Under no circumstances should such information be encoded in the respective messages. Instead, the message delivered over the push notification service does not contain any information. Rather, it only contains a hash, a random or secret number and is therefore not predictable. The app now polls the backend for test results or whether the user is at risk. This request includes the random number sent in the push notification, which is used by the backend to look up the message for the response or answer with a dummy message. Note that neither the request nor corresponding response are visible to the push notification service.

However, the simple presence of a message can also leak information about health status to the service. For instance, if a notification (of a certain format) is only sent to users that are positively tested, the service knows that a user receiving a notification is infected with Covid-19. For this scenario, all users that have been tested—positive or negative—can receive such a notification. With current German numbers of up to 7,000 new infections per day and around 50,000 daily tests, only 7/50 of users receiving such a notification are actually positively tested. To achieve a sufficient obfuscation of information on health status, for each notification sent to a user tested positively on Covid-19 and for each notification sent to users at risk, 999 notifications are sent to randomly selected apps. All apps query the backend, which looks up the most recent data for the respective app and sends the answer. Thus, only 1 out of 1000 notified apps actually receives information in the response from the backend.

Hence, using push notification services is required to provide a scalable and efficient system and overcome technical limitations. By transmitting random messages to apps and hiding the relevant messages in additional noise, the operator of the push notification service cannot infer any sensitive information from observing the notifications.

6 Discussion of Security Goals

In this section, we analyze the properties achieved by the PEPP-PT design and point out residual risks.

NF-REQ 1 Prevent Sybil attacks (attacker registering many accounts at the backend)

To regulate user registration, the system uses Captchas to require human interaction and proof-of-work, a computation that takes at least duration *D* to solve to any adversary, with an energy/computation cost *C*. One proof-of-work is needed per account registration. To register *N* accounts, the adversary has to spend *N***D* time and *N***C* cost. If *D* and *C* are chosen properly, this de-incentivizes the registration of many accounts.

NF-REQ 2 Temporary identifiers (EBID) received by the backend must be authentic

This goal is currently addressed with authenticated channels between backend and apps. Only the backend is in possession of the secret keys to compute valid EBIDs.

F-REQ-7 Legitimate not-at-risk users of the system should not be falsely notified to be "at risk".

To falsely notify a not-at-risk user, a valid EBID in the relevant time frame must be inserted into the proximity history of a Covid-19 positive person. Faking valid EBIDs is not possible (see above), but recording them over Bluetooth and replaying them is possible. The scenario would be a tech-savvy A2 adversary who collaborates with another A2 adversary who provides previously recorded valid EBIDs (note that a single adversary recording and replaying EBIDs has actually been in the physical vicinity and, thus, would not report false EBIDs). These recorded EBIDs can then be inserted into a proximity history of the attacker. Only if the attacker is then tested Covid-19 positive within the next 21 days, would the history—including fake EBIDs—be uploaded to the backend. In addition, the risk scoring algorithm in the backend would need to determine the contact with the fake EBID as relevant. In that case, a not-at-risk-user would be notified.

The risk of this attack is considered to be from low to negligible. It would require collaborating A2 adversaries, one of whom must become a Covid-19 patient within the next 21 days.

Alternatively, a single attacker could collect a valid EBID of its victim and start broadcasting it without being close to the victim. For the attack to succeed, another person scans the EBID for a sufficiently long time in close proximity and the other person is tested positive in the next 21 days. However, this attack imposes a threat to the attacker's health as he has to be close to an infected person for a long time and is therefore at risk of being infected. The risk for the attacker exceeds the value of the attack.

F-REQ-8 Legitimate at-risk users of the system should be notified about being at risk

The discussion is similar to the one above, except that no collaboration between adversaries is needed. An A2 adversary can edit their proximity history and remove EBIDs before they are uploaded to the backend, in case the adversary is Covid-19 positive tested. Given the fact that uploading the proximity history is voluntary anyway and that users are free to not report any contacts at all, this attack is not considered relevant.

F-REQ-9 Only legitimate users diagnosed with the infection should be able to upload their proximity history to the backend (authorization/authentication).

This goal is met based on the secure TAN procedure.

F-REQ-10 Only authorized personnel (i.e., backend administrators) should be able to access information stored in the backend

It is assumed that the backend operator follows security best practices in terms of identity and access management for backend resources. A detailed security concept for backend operators is out of the scope of this document.

F-REQ-11 Only authorized personnel (i.e., health authority officials and the backend administrator) can validate a TAN

This goal is met since only the health authority obtains the necessary credentials to validate TANs.

NF-REQ 8 The pseudonyms of participants (infected or not) should not be linkable to long-term identifiers

A1, A2, and A5 adversaries can only observe EBIDs and do not have access to the permanent PUID pseudonym. EBIDs are not linkable to each other and linking an EBID to its PUID requires knowledge of the secret key of the respective time interval. This secret key is stored in the backend (ideally in an HSM). That is, an A4 adversary would be able to map an EBID to its PUID, as long as the secret key for the EBID's time frame is still available (i.e., less than 21 days old). In fact, this is the main purpose of the backend.

However, it should be noted that users can change their PUID at any time—without the backend noticing—by simply re-installing the app. Therefore, they can change their long-term pseudonym at will. Further, no other personal data apart from the PUID pseudonym are stored or processed by the backend.

NF-REQ 9 Infected users should be personally identifiable only by the health authority

Covid-19 positive users never upload personal data when uploading the proximity history. However, by observing network traffic, an A5 adversary can determine that a user is uploading large chunks of data to an endpoint of the PEPP-PT app and conclude that the sender has been Covid-19 tested. This applies to ISPs, network operators, but also to hackers setting up rogue access points or sniffing public WiFi networks. This attack is technically not difficult to carry out, but must take place while the user is running the TAN workflow and talking on the phone with health authorities (i.e., it is less likely that the user will use a public WiFi at this point). A malicious ISP could also use other means to determine that a user has been Covid-19 infected; for instance, tracing phone calls from the health authorities.

Nevertheless, there is a residual risk that a determined A5 attacker might be able to observe traffic that indicates a Covid-19 infection. Mapping that traffic to a real person would be another challenge, but the (low) risk remains.

A mitigation would be the use of mix networks such as Tor. They are not an inherent part of this concept, but users are free and encouraged to use them if they do not wish to accept the residual risk.

NF-REQ 10 PII of the at-risk people should not be learned by anyone

This goal is met as all users have the same behavior regarding the reception of at-risk notifications. In particular, all users (at-risk and not-at-risk users) regularly request updates of the risk score in the backend. The answers have the same format, no matter whether the user is at risk or not. Therefore, if the backend and the user are honest and—since their communication channel is confidential (and leaks no metadata)—even an eavesdropper cannot distinguish at-risk users from not-at-risk users.

NF-REQ 11 The location privacy of users should be preserved

The system does not collect any specific location information. However, it does collect EBIDs and uploads them to the backend in case of a positive Covid-19 test. As the proximity history will contain only PEPP-PT EBIDs (especially no UUIDs of BLE beacons with fixed known locations), no mapping between the uploaded UUIDs and known locations exists.

NF-REQ 12 Co-locations/partial social graph of not-infected users shall not be exposed to unauthorized parties

CTD contains time information. As a result, the backend learns the co-locations/partial social graph of identifiers. We note that only the backend processes aggregated CTD information. Therefore, if this entity is trusted then other unauthorized parties do not learn co-location information or social graphs through the data exchanged in the app.

7 Future mitigations

The proposed system aims for a pragmatic approach under the given circumstances. We acknowledge that the security of the system can be gradually improved by further means, some of which we list here:

Trusted Execution Environments at the server side

Code running on the server must be trusted in the sense that it correctly notifies only those users that have been determined to be at risk of a Covid-19 infection. Further, although it has no access to personal information other than the PUID pseudonym (which can be changed by the user at any time by re-installing the app), skeptical users might suspect that the backend collects more information or tries to uncover the real identity of users by data analytics and linking information with external knowledge. Publication and auditing of the source code of the backend is reasonable. However, it might not be enough to convince skeptical users, as they are not given a guarantee that the backend actually runs the published source code. To further convince users that the code running in the backend is actually the one that has been released and inspected, remote attestation mechanisms and Enclaves as Trusted Execution Environments (TEE) may be used.

This may require a significant change in the backend architecture and it must be considered that remote attestation of Intel SGX Enclaves should not rely on the availability of the Intel Attestation Service (IAS) but should rather use the "DCAP" option[3]. Nevertheless, it is a viable option to increase trust in the backend at a firm technical basis.

Mix networks to avoid traffic analysis

In the current concept, a determined A5 adversary can learn from network traffic observations that a user is uploading the proximity history and might, thus, be Covid-19 infected. Although we consider this attack to impose a low risk because only ISPs could carry it out at a larger basis, the use of mix

[3] https://software.intel.com/en-us/sgx/attestation-services

networks (mixnets) would be a reliable countermeasure. By means of mixnets [10] and onion routing networks such as Tor[4], the communication endpoints are disguised and it is (even for A7 adversaries) difficult-to-impossible to infer information about the communication behavior of individual users. The concept does not exclude the use of such mixnets. Therefore, users are free to use them if they wish to do so. However, integrating mixnets or onion routing in the core architecture of any large-scale system to run all user's communication over the mixnet/onion routing networks is anything between "non-trivial" and "impossible". Tor, the world's largest onion routing network, currently has 2–2.5 million daily active users. This is less than 5% of the scale that PEPP-PT plans.

Signed EBIDs including timestamps

Currently, EBIDs are created by the backend with a time-specific global secret key and the user's PUID. An attacker can record and replay these EBIDs and modify the corresponding timestamps at will, as long as they remain in the validity windows of the corresponding key (e.g., one hour). This would allow an A2 attacker to turn a short encounter with a user (that would not be relevant for a Covid-19 infection) into a longer encounter that might be deemed as relevant by the risk scoring algorithm. As a mitigation, users should create signed EBIDs by themselves: calculating an HMAC of the current timestamp and their PUID using a secret time specific key. This will prevent an attacker from faking timestamps for EBIDs. However, it raises scalability challenges and would impede the use of an HSM for key storage in the backend. Nevertheless, it is a viable improvement that is currently being evaluated as a joint French–German collaboration to further unify Covid-19 tracing architectures across Europe.

[4] https://www.torproject.org

8 Glossary

BK	Broadcast Key
EBID	Ephemeral Bluetooth ID
ECC	Encrypted Country Code
HMAC	Hash-based Message Authentication Code
IAM	Identity & Access Management
PHS	Public Health Service
PID	Push Notification ID
PII	Personally Identifiable Information.
PoW	Proof of Work
PUID	Pseudonymous User ID.
RPA	Resolvable Private Address
TAN	Transaction Authentication Number

9 References

[1] Institut National de Recherche en Informatique et en Automatique (INRIA), Fraunhofer, „ROBERT - ROBust and privacy-preserving proximity Tracing protocol,“ 16 April 2020. [Online]. Available: https://github.com/ROBERT-proximity-tracing.

[2] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, M. Parker, D. Bonsall und C. Fraser, „Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing,“ *Science,* 31 March 2020.

[3] RFC6749, „The OAuth 2.0 Authorization Framework“.

[4] M. Jones, *RFC 7797 - JSON Web Signature (JWS) Unencoded Payload Option,* 2016.

[5] RFC7914, „The scrypt Password-Based Key Derivation Function,“ 2016.

[6] A. Biryukov, D. Dinu, D. Khovratovich und S. Josefsson, „IRTF Internet Draft - The memory-hard Argon2 password hash and proof-of-work function,“ March 2020. [Online]. Available: https://tools.ietf.org/html/draft-irtf-cfrg-argon2-10.

[7] nano Currency, „nano Proof of Work,“ [Online]. Available: https://github.com/nanocurrency/nano-pow.

[8] National Institute of Standards and Technology (NIST), *Guide to Bluetooth Security,* 2017.

[9] PEPP-PT Consortium, *Stopp Corona App German TAN System,* 2020.

[10] D. Chaum, „Untraceable electronic mail, return addresses, and digital pseudonyms,“ *Communications of the ACM,* p. 84–90, 24 February 1981.

[11] Robert-Koch-Institut, „Aktuelle Statistik meldepflichtiger Infektionskrankheiten," *Epidemiologisches Bulletin 12/2 - Aktuelle Daten und Informationen zu Infektionskrankheiten und Public Health,* pp. 9-11, 19 March 2020.