Pan-European Privacy-Preserving Proximity Tracing

Proximity Measurement Specification

Status: 23.4.2020

Executive Summary

The SARS-CoV-2 virus is mainly spread via droplet infection. Current epidemiological models assume a distance of less than 2 meters over a time of 15 minutes that puts people at an increased risk of infection. The duration and type of interaction between people also has a high impact on the infection risk, but is very difficult to measure using the measurement tools chosen.

Measuring the proximity to other persons while both persons do not know whether they are infected is a key requirement of proximity tracing systems. Today, this can be achieved with Bluetooth Low Energy (BLE) technology that is built in standard smartphone hardware. Although BLE has been built to transmit data, the radio signal strength can be used to measure proximity.

Contents

1 Introduction Bluetooth Low Energy Concept

PEPP-PT requires scanning the environment via Bluetooth Low Energy (BLE). The PEPP-PT App shall advertise and scan. To address privacy and protect from creating movement profiles, the PEPP-PT app will use a pool of ephemeral Bluetooth IDs (EBIDs), which are cryptographically generated at PEPP-PT backend side, fetched by the PEPP-PT application, and that will be used in the payload part of the PEPP-PT advertisement. The EBIDs change over time and will be renewed once the pool is empty. The IDs will never be re-used. Once a pool is empty, a new pool of EBIDs will be fetched from the backend. During scanning, the PEPP-PT app will filter by 16 Bit service UUIDs. All proximity data will be stored locally on the device for a maximum of 21 days. All data that is older than 21 days shall be permanently deleted from the device. If the data shall be uploaded to the PEPP-PT backend, the user must give personal consent. The upload of the stored data is voluntary and must not be uploaded without a verified positive test for CoVID-19.

2 Description of Functionality

2.1 EBID Pool

The EBIDs contained in the pool are generated on server side using cryptographic keys that have a limited validity; in addition, the EBIDs have a limited lifetime, for example 1 hour. Thus, each EBID has a limited lifetime as well, will be deleted from the device once it is expired, independent from former usage. The pool is securely transferred to the mobile application, delivering EBID and validity period for each EBID.

2.2 Local Device Storage

Proximity information data is kept encrypted on the device for a maximum of 21 days. Any data older than 21 days will be overwritten by new data. Given no new data is collected, the proximity data older than 21 days will be deleted.

Proximity data consists of an EBID from a scanned device, the phone model of the scanned device as identified by the manufacturer ID, the date and time of the proximity detection, the TX power read from the BLE advertisement, the RSSI value. In addition to that, corrections to the RSSI value are used.

2.3 Tx / Rx Compensation Values

There are two types of RSS corrections: static and dynamic corrections. The source of the corrections are the sender and the receiver.

Static compensation values are handset model specific deviations in Bluetooth hardware and material and shape of the phone body. Static compensation values will be made available via public databases.

Device state information may be used to dynamically adjust the static handset model specific calibration, which results in a dynamic tx / rx compensation value.

The data format above provides 1 byte [-127, 126] compensation values to compensate the received RSSI to a reference with a dB offset. Such offset, herein referred to as "compensation value" obtains the static handset model specific calibration value and, optionally, a dynamic calibration value based on the current state of the device. Such dynamic value may be added to the static one to result in a new compensation value to be uploaded to the server. There are two compensation values:

- Tx compensation value: the compensation value of the device transmitting the advertisement, the BLE peripheral. The Tx compensation value has to be broadcasted with the advertisement, e.g. in the manufacturer payload section, to be recognized and stored by the central devices building their proximity history.
- Rx compensation value: the compensation value of the device receiving the advertisement, the BLE central.

Since the Bluetooth Rx levels show a strong dependency on the phone being carried in a pocket or currently used for calls or browsing in the palm of the user's hands, it is recommended to include auxiliary measures into the proximity detection. Such measures indicate the current phone position / usage and thus help to distinguish indoor / outdoor and other usage scenarios.

For data privacy reasons such measures shall not be stored or transmitted in plain form, but rather incorporated in one calibration value as explained above.

Device state information that may be taken into account to further improve the proximity estimation, may include

- **WiFi state**: The WiFi may be switched **off**, **on**, or the device may even be **connected** to some WiFi Access point.
- **Display state:** The display may be on or off. If the display is on, the phone is very likely used outside a pocket, e.g. for browsing apps or currently being charged.
- **Proximity sensor**
- **Light sensor**
- **Gyroscope sensor** phone tilt indicating a horizontal vs vertical position of the phone
- **Charging state**
- **Acceleration sensor**

The protocol and data format described in the document are agnostic of any implementation details regarding static and dynamic calibration. If no static or dynamic calibration is implemented, the corresponding contribution to the compensation value is zero and thus the compensation value is not changed.
If no compensation for a specific model is available at all, the corresponding compensation value is equal to zero.

3 Configuration of Peripheral and Central

Each PEPP-PT application shall act as a peripheral (sender) and a central (receiver) at the same time.

The peripheral broadcasts the specified advertisement data. The central discovers nearby BLE devices, and extracts relevant information from the received peripheral broadcasts.

3.1 Peripheral

The peripheral shall broadcast the advertisement continuously (if possible). The peripheral data shall be generated such that it follows the PEPP-PT advertisement format. It shall include:

- Flags:
  - The advertisement type shall be connectable: Fixed value
- Manufacturer data:
  - Manufacturer identifier: As provided by the device's OS platform
  - Manufacturer specific data: Handset dependent value for path loss correction (1 byte)
- Service data:
  - UUID: 2-byte service identifier; ideally as 16-bit UUID registered with Bluetooth SIG.
  - Payload: Temporary BLE identifier, i.e. EBID, which changes over time following a rotation scheme (16 byte)
- Tx power level:
  - Tx power shall be included in the advertisement data

3.1.1 PEPP-PT Advertisement Format

The used BLE advertisement format shall conform to the format given in the Table below.

Table 1 - Advertisement format

Data Type	**Length**	**Value**	**Details** **2 byte type + payload**
0x01 (Flags)	2 bytes	0x02	1 byte, set by OS, i.e. not connectable
0x0A (Tx power)	2 bytes	0x0C	1 byte, set by OS, for proximity estimation
0xFF (Manufacturer data)	3 (+ 1) bytes	e.g. 0x004C (Apple) Hardware correction factor	2 byte manufacturer ID as provided by the OS 1 byte custom payload
0x16 (Service data)	Up to 20 bytes	0x8531 (example) 0x0102…16 (EBID)	2 byte (service data UUID) up to 16 bytes custom payload

This format has the advantage of providing a large amount of custom payload which shall be used as follows:

- 1 byte for the hardware correction factor as part of the manufacturer data,

- 16 bytes for PEPP-PT EBIDs as part of the service data payload. These EBIDs shall be changed over time, for example in 15-minute intervals

PEPP-PT would register a 16 Bit UUID for the service data with Bluetooth SIG if required or advisable.

3.2 Central

The central shall turn on the BLE scanner and scan continuously (if possible). In addition, a scan filter to restrict scan results to only those that are of interest to them shall be implemented with the following configuration:

- Manufacturer data:
  - Description: Scan for any manufacturer.
- Service UUID:
  - Description: Set filter to restrict on PEPP-PT UUID

Having implemented the above settings and filters, the relevant data can be extracted from detected advertisements, including:

- Manufacturer identifier and hardware path loss correction (Manufacturer payload, 1 byte)
- Service data UUID (filtered)
- Temporary BLE identifier, i.e. PEPP-PT EBID, (payload, 16 byte)
- Tx power (of peripheral, 1 byte)
- Received signal strength indication (RSSI) in dBm

The data extracted from scanned advertisements should be stored on the device and shall be limited to:

- The local temporary BLE identifier, i.e. PEPP-PT EBID, as used while advertising as Peripheral.
- Temporary BLE identifier, i.e. PEPP-PT EBID, as read from the advertisement
- Tx power (of peripheral)
- Received signal strength indication (RSSI) in dBm
- Time of the proximity detection

4 Proximity History Data Format

The following section describes the data format used for uploading the proximity history to a server to process the risk computation and trigger the warning system.

4.1 Naming Conventions

device A	The device running the app instance measuring and storing the data.
device B	A device in the BT range of device A, so A is receiving the advertisement from B. There may be multiple devices B (B1, B2, … Bn) in the proximity of the user.

EBID	Ephemeral Bluetooth ID broadcasted by a PEPP-PT device as BLE advertisement (peripheral).

4.2 Proximity History

Key	Data type / example	Content
header	PEPP_PT_HEADER	See section 4.3
bt-scans	JSON Array of BLUETOOTH_SCAN_LIST	See section 4.4

4.3 Data Type: PEPP-PT_HEADER

Each data upload shall contain a header with meta information such that the background processing can derive appropriate actions.

The examples below have been derived from a Samsung Galaxy S10+ and an iPhone 11.

Sine some header values contain text defined by the device, proper JSON escaping or conversion in base64 has to be implemented.

Key	Data type / example	Content
dt	32-bit timestamp Unix epoch time in seconds resolution	timestamp of upload
ver	"1"	Header data format version
id	16-Octet-UTF8 "01020304-0506-0708-090A-0B0C0D0E0F10"	Current **EBID** of device A
tx-gain	number	Static handset model specific tx-gain of device A (default: 0)
rx-gain	number	Static handset model specific rx-gain of device A (default: 0)
app-pn	"com.pepppt.sample"	app package name Android: pm.PackageInfo.packagename iOS: Bundle ID The package name here is just an example. The final one may differ.
app-version	"1.0"	app version Android: pm.PackageInfo.versionName

		iOS:
platform	“android” or “ios”	device OS version (Android / iOS)
device-manufacturer	“samsung” “apple”	device manufacturer Android: os.Build.MANUFACTURER iOS: “apple”
device-model	“SM-G920F” “iPhone11,6”	device model Android: os.Build.MODEL iOS:
os-version	“10.0” “iOS 13.4”	device OS version Android: os.Build.VERSION.RELEASE
Android specific		
android-app-vc	“101”	pm.PackageInfo.versionCode
android-os-core	“29”	os.Build.VERSION.SDK_INT
android-os-ink	“G975FXXS4BTB3”	os.Build.VERSION.incremental
android-os-id	“QP1A.190711.020”	os.Build.ID
android-os-hardware	“exynos9820”	os.Build.HARDWARE
android-os-bootloader	“G975FXXS4BTB3”	os.Build.Bootloader
android-osboard	“exynos9820"	os.Build.Board
android-os-radio	“G975FXXU4BTAA,G975FXXU4BTAA”	os.Build.getRadioVersion() The example shows a Galaxy S10+ dual SIM, so 2 radio firmware versions.
iOS specific		
ios-model-id	“D331pAP”	
ios-os-build	“17E255”	
ios-os-core	“Darwin 19.4.0”	

4.4 Data Type: BLUETOOTH_SCAN_LIST

Please note that it is recommended to store the app’s version with each snapshot since the version may change over time and measurement data from different app versions may be transmitted in case of an upload of the proximity history. In case we need to consider the app version when evaluating the

Key	Data type	Content
dt	32-bit timestamp Unix epoch time in seconds resolution	Time of the reception of BT scan
v	"4"	BLUETOOTH_SCAN_LIST data format version
av	e.g. "1.0"	App-version Android: pm.PackageInfo.versionName iOS:
sci	32 bit integer in hexadecimal format without leading zeros e.g. "3E8" for 1.000ms	BT snapshot scanning interval in milliseconds Time interval between two BT scans (e.g. 1.000 = 1s)
agi	32 bit integer in hexadecimal format without leading zeros e.g. "2710" for 10.000ms	BT snapshot aggregation interval in milliseconds Aggregation time for BT scans (=Time interval between two aggregated BT scans) (e.g. 10.000 = 10s)
rxc	1 byte [-127,126] dB Rx RSSI or proximity compensation value of device A during the current interval	The Rx compensation value is the sum of • Static Rx gain of device A • Optional dynamic Rx gain of device A based on current IMU data, which may be added to the client-side BLE scan ("central") algorithm
bts	JSON array of BLUETOOTH_SCAN	

4.5 Data Type: BLUETOOTH_SCAN

Key	Data type	Comment
id	16-Octet-UTF8	Current **EBID** of device B 96 EBID as 16-Octet-UTF8
mf	16 bit	Manufacturer ID of BLE advertisement of device B
rm	1 Byte	**Mean RSSI** [-127, 126] https://developer.android.com/reference/android/bluetooth/le/ScanResult#getRssi
rv	1 Byte	**RSSI Variance**

ct	1 Byte	**Count of observations of this EBID [0, 255]** (Number of Received Packages of this EBID)
tp	1 Byte	**Mean Tx power level in dBm** [-127, 126] https://developer.android.com/reference/android/bluetooth/le/ScanResult#getTxPower
tc	1 Byte	**Tx RSSI / proximity compensation value of device B [-127,126] dB** The Tx compensation value is the sum of • Static Tx gain of device B • Optional dynamic Tx gain of device B based on IMU data. Which may be added to the client-side peripheral algorithm.