HAL open science

On using Bluetooth-Low-Energy for contact tracing

Mathieu Cunche, Antoine Boutet, Claude Castelluccia, Cédric Lauradoux, Vincent Roca

To cite this version:

Mathieu Cunche, Antoine Boutet, Claude Castelluccia, Cédric Lauradoux, Vincent Roca. On using Bluetooth-Low-Energy for contact tracing. [Research Report] Inria Grenoble Rhône-Alpes; INSA de Lyon. 2020. hal-02878346v5

HAL Id: hal-02878346

https://inria.hal.science/hal-02878346v5

Submitted on 1 Sep 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On using Bluetooth-Low-Energy for contact tracing

Mathieu Cunche[†] [*]
Antoine Boutet[†]
Claude Castelluccia
Cedric Lauradoux
Vincent Roca

PRIVATICS Team, Inria, France
[†]INSA-Lyon, CITI Lab, France
desire-contact@inria.fr

September 1, 2020
v1.3

1 Introduction

The goal of contact tracing is to keep track of contact between individuals in order to inform persons who have been exposed to a risk of infection. Performing contact tracing with the aid of digital tools such as smartphones has been considered by a number of countries. Many of those contact-tracing proposals are based on Bluetooth Low Energy (BLE), a medium range wireless technology available on most smartphones. The core idea is to use BLE to exchange information between nearby smartphones in order to keep track of the contact of their owners.

Even if it has not been designed to measure distance, its limited range along with signal strength indicator could provide a good enough estimation of proximity [13]. Furthermore, BLE has been designed to have a low energy footprint, which is important for a task that must be kept running all the time on a device with limited battery resources.

Behind the consideration of distance estimation and energy consumption remains several issues regarding the technical choices for the device-to-device communications. This document will cover those details and will discuss how contact tracing can be implemented based on BLE.

This document solely focuses on BLE communication aspect of a contact tracing application, in particular, it does not discuss in detail the distance estimation nor the energy consumption aspects. The goal of this document is to introduce the technical elements behind BLE-based contact tracing, present the technical limitations and describe the envisioned solutions.

[*]Mathieu Cunche is the contact author. The other co-authors are listed in alphabetical order.

1.1 Requirements for a contact tracing application

To be efficient, a BLE-based contact tracing application has to satisfy a number of properties for the functionality and privacy aspects. We propose the following list that covers the main requirements that must be fulfilled by a contact tracing application:

[DATA_TRANSMIT] : a device running the app can transmit pieces of data to nearby devices running the app.

[DATA_RECEIVE] : a device running the app can receive pieces of data from nearby devices running the app.

[ALWAYS_ACTIVE] : the transmission and reception of data must always be active while the app is running.

[NO_TRACKING] : the pieces of data transmitted by the app must not be useful for physical tracking. In other words, an attacker should not be able to leverage contact tracing data to physically track a user.

[ADDR_RAND] : data transmission should not interfere with address randomization scheme used in BLE.

The requirements DATA_TRANSMIT, DATA_RECEIVE and ALWAYS_ACTIVE cover functionality aspects while NO_TRACKING and ADDR_RAND cover privacy aspects.

2 Bluetooth Low Energy

Bluetooth Low Energy is a lightweight subset of Bluetooth that has been introduced in 2010 as part of the Bluetooth specification version 4.0 [5, Vol 6]. As the legacy Bluetooth, it operates in the 2.4GHz ISM band, in which are defined 3 advertisement channels and 37 data channels.

In BLE, devices can endorse one of two role, *peripheral* or *central*, and are identified by a *device address*. Once connected, a peripheral and a central can communicate by exchanging packets that over the BLE data channels.

2.1 Advertising

As a preliminary step to establish a connection, BLE features an advertising scheme in which peripherals announce themselves by periodically broadcasting *advertising packets* over the three advertisement channels. Central devices detect nearby devices by tuning on the advertisement channel and listening to *advertising packets*.

An *advertising packets* includes an *Advertising Address* followed by some *advertising data* (see example in Figure 1). The Advertising Address is 48 bits long and corresponds to the device address of the emitting peripheral. The advertising data can contain up to 31 bytes organised in Advertising Data Elements, type-length-value structures that carry information such as the device name, service UUIDs, service data or manufacturer specific data[1].

2.2 Custom applications of BLE advertising

The primary purpose of BLE advertising is the discovery of nearby devices with their characteristics, prior to a connection. However, the freedom left by technical specification offers room to other purposes for instance the transmission of data without a connection [11]. BLE *Beacons* are

[1] `https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile`

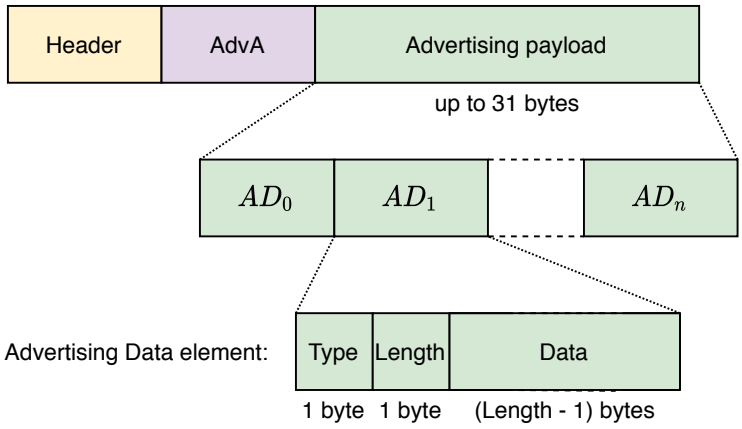

Figure 1: A BLE advertising packet includes the device address of the emitter (AdvA), along with an advertising payload organised in Advertising Data Elements.

an example of such application which are static emitters broadcasting a unique identifier used for geolocation purposes. BLE beacons are based on BLE advertising and use the advertising packets to carry their identifier in a dedicated field[2]. Furthermore, major Vendors, such as Apple, Google and Microsoft, have developed close range device-to-device communication framework based on BLE advertising [9, 10]. The advertising packets are used to carry datagrams for a wide range of applications: device pairing, task and file transfer, device tracking, etc. [4, 10].

2.3 Device address randomization

As a feature to protect users against tracking, BLE supports address randomization. More specifically, the Bluetooth specification defines *private addresses* which are randomly generated addresses that are to be rotated periodically. This feature is used in many devices, including smartphones running the most recent versions of Android and iOS.

2.4 ATT: the attribute protocol

The attribute protocol (ATT) is a protocol included in BLE for discovering, reading, and writing attributes on a remote device. Attributes are composed of a handle, a type and a value, and are generally exposed and organized in a GATT profile (Generic Attribute Profile).

The ATT protocol requires a connection between two devices, but this connection does not necessarily require pairing and can be done without authentication. As a result, two devices can use the ATT protocol to exchange information opportunistically and without user intervention.

3 Models for contact tracing with BLE

Recent developments of BLE-based contact tracing have seen the adoption of two models: a connected model in which two devices establish a connection to exchange data, a broadcast model in which devices broadcast and collect undirected messages. This section presents the two approaches, after having introduced common features.

3.1 General features shared by all models

In order to identify a contact person, current proposals rely on the transmission of contact tracing data, which include an identifier as well as metadata. Identifiers are rotated periodically

[2] `https://developer.apple.com/ibeacon/`

and are unlinkable in order to protect against tracking (cf. NO_TRACKING requirement), and other kinds of attacks (e.g., social graph reconstruction).

The metadata accompanying the identifier can include a protocol version identifier, a country code for roaming , and information about the transmitter characteristics to improve the accuracy of proximity estimation.

3.2 Broadcast model

The broadcast model leverages the advertising mechanism of BLE to carry contact tracing data in advertising packets. Each device periodically broadcasts advertising packets containing the contact tracing data carried within an advertising data element. Simultaneously, each device collects incoming advertising packets, and extracts and records the contact tracing data (see Figure 2).

Advertising packets have a payload of 31 bytes but 3 bytes must be reserved for the Flags AD structure, leaving 28 bytes for the contact tracing data. Furthermore, each AD structure starts with a 2 bytes long type-length header, further reducing the available space. If only one AD structure is used (in addition to the Flags AD structure) there is a total of 26 bytes that can be used to carry contact tracing data.

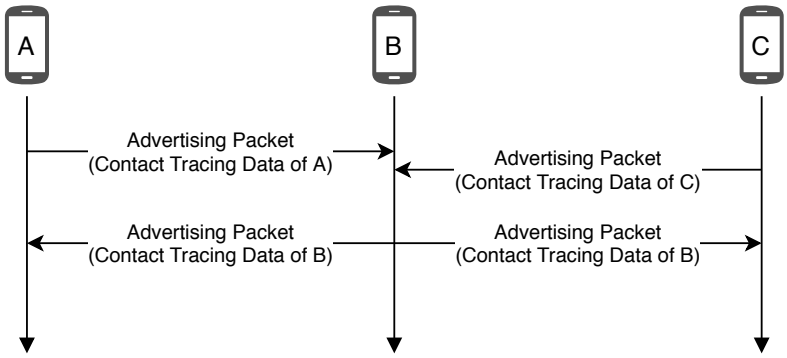

Figure 2: Broadcast model for BLE-based contact tracing. Each device broadcasts advertising packets carrying its contact tracing data. An advertising packet can be received by several devices in range (e.g., the advertising packet of B is received by A and B).

Several types of AD element can be used to carry the contact tracing data. A first option is to use the service data associated to a dedicated service. This first option requires to advertise the service itself in addition to the service data, leaving only 20 bytes for the contact tracing data.

A second option is to use the manufacturer specific AD structure which is designed to carry custom data. In this case, a single AD element is used, limiting the AD header overhead to 2 bytes. A manufacturer specific AD must start with a company ID on 2 bytes, leaving 25 bytes for contact tracing data.

As of today, most systems have adopted the first option: 20 bytes of contact tracing data carried by a service data AD element.

3.3 Connected model

The connected model relies on the connection between two devices to exchange the exchange of contact tracing data (see Figure 3). Each device advertise its support of the contact tracing system by including the corresponding service UUID in the advertising packets. Upon detection of a nearby device supporting the service , a device will establish a connection and will exchange contact tracing data with the remote device.

Most solutions exchange data using the ATT profile: each device expose a service and the contact tracing data is transferred by writing and reading attributes of this service on the remote device. The amount of data that can be exchanged by two devices in the `Connected Model` is only limited by the duration of the connection.

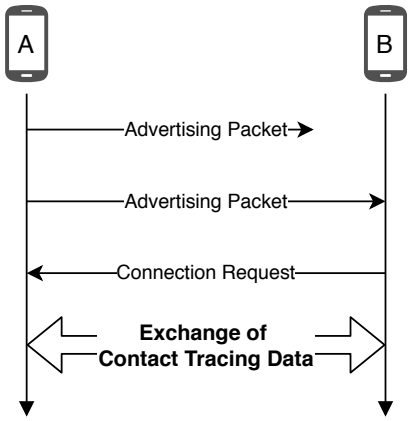

Figure 3: Connected model for BLE-based contact tracing. Device A announces itself by broadcasting advertising packets including the UUID of the contact tracing service. After receiving one of those packets, device B initiates a connection with B, then A and B exchange their contact tracing data through this connection.

3.4 Hybrid model

The broadcast and connected models are not mutually exclusive and can be used simultaneously. An hybrid model combining the two approaches can be envisioned (see Figure 4). Such hybrid model has been considered [17, 12] to overcome some constraints imposed by iOS[3].

In the hybrid model, devices that do not support the broadcast model will use the connected model to exchange contact tracing data; other devices will use the broadcast model.

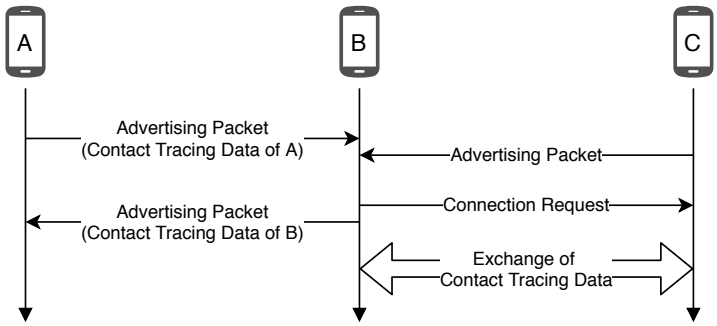

Figure 4: Hybrid model for BLE-based contact tracing. Device A and B exchange their contact tracing data following the broadcast model. Due to limitations of the OS, device C is unable to use the broadcast model. Therefore, device B and C will use the connected model to exchange their contact tracing data.

[3]"the protocol uses both broadcast-oriented and connection-oriented BLE modes"`https://github.com/TCNCoalition/TCN`

3.5 Comparison of the communication models

The broadcast and connected models each have advantages and limitations. First with regard to the amount of data, the broadcast model is limited to the capacity of an advertising packet which offers at most 25 bytes of useful data; while in the connected model the amount of data that can be exchanged is only limited by the bandwidth and duration of the connection.

A second aspect that differs between the two models is the scalability. In the broadcast model, a single advertising packet is potentially enough to transmit contact tracing data to all nearby devices; whereas with the connected model, it is necessary to establish a connection for each nearby device. Therefore, the broadcast model is more scalable than the connected model.

3.6 Advertising parameters

A key requirement of both models is that nearby devices receive advertising packets of each other. In the broadcast model, those packets carry the contact tracing information, while in the connected model, they enable the detection and identification of a device preliminary to the creation of the connection used to exchange contact tracing data.

The reception of advertising packet depends on several parameters on both the advertiser (emitter of the advertising packet) and the scanner (receiver of the advertising packet). More particularly, for a successful reception, the scanner must be in receiving mode when the advertiser transmits the advertising packet.

The advertiser periodically transmits advertising packets with a period called the `advertising interval`. The scanner, packet is not continuously scanning and its scanning behavior is characterized by two main parameters: the scanInterval, which determine the time between two consecutive scans, and the scanWindow, which determine the duration of a scan (see Figure 5).

Those parameters have a direct impact on the discovery speed and on the energy consumption. Depending on the application various advertising and scan parameters can be configured. For instance Android supports a low power scan mode and a low latency scan mode, in which the default[4] values of scanInterval and scanWindow of 512 ms and 5120 ms, respectively 4096 ms and 4096 ms[5].

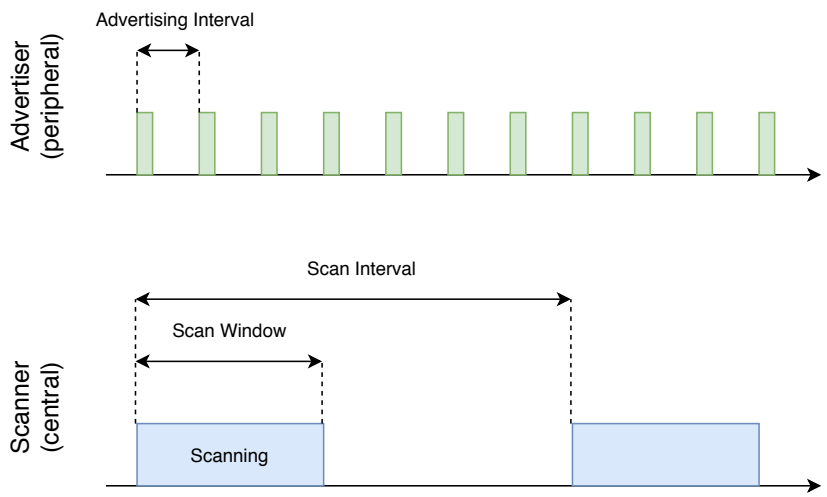

Figure 5: Simplified representation of advertising and scan behaviours in BLE.

[4]Those values can vary from one device to the other because of OS customization by vendors.

[5]`https://android.googlesource.com/platform/packages/apps/Bluetooth/+/master/src/com/android/bluetooth/gatt/ScanManager.java`

3.7 BLE advertising payload

In the `Broadcast Model`, contact tracing data is carried in the payload of advertising packets. As discussed in section 3.2 there are several options on how to format this data. In this section we present the details of the advertising payload of two prominent solutions: the GAEN and StopCovid, the french contact tracing application.

3.7.1 Payload of the Google & Apple Exposure Notification framework

The GAEN (Google & Apple Exposure Notification) framework, is a technical solution for contact tracing introduced by Apple and Google [1]. In GAEN, the contact tracing data is carried by a `Service Data` field associated with a service dedicated to contact tracing: the *Exposure Notification* Service identified by the UUID `0xFD6F`. The contact tracing data is composed of a 16 bytes identifier, called `Rolling Proximity Identifier`, accompanied by 4 bytes of encrypted metadata, called `Associated Encrypted Metadata` (AEM)[6]. The AEM is composed of a service version on 1 byte, transmit power level indicator on 1 byte, and 2 additional bytes reserved for future use. The GAENspecifications [1] defines the payloads of advertising packets (see Figure 6). It includes:

- `Flags` (3 bytes): Not specific to exposure notification, but must be included in every `advertising packet`.
- `Complete 16-bit Service UUID` (4 bytes): carry the UUID of the exposure notification service (`0xFD6F`).
- `Service Data - 16-bit UUID` (24 bytes): carry the data associated to the exposure notification service along with the the UUID of the exposure notification service.
  - the exposure notification service UUID (`0xFD6F`)
  - `Rolling Proximity Identifier` (16 bytes): the temporary identifier derived by the EN framework.
  - `Associated Encrypted Metadata` (4 bytes): encrypted metadata (version, transmit power level) associated to the device.

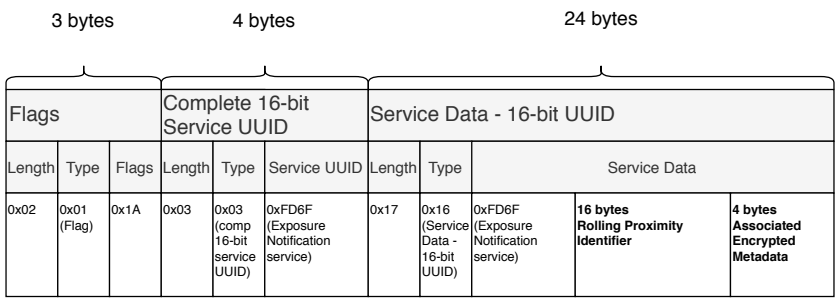

Figure 6: Payload of the advertising packet as specified for the GAEN framework. The `Rolling Proximity Identifier` and the `Associated Encrypted Metadata` are carried by a `Service Data` structure associated to the Exposure Notification service (UUID `0xFD6F`)

[6]The AEM is encrypted using a key derived from the Temporary Exposure Key (TEK - the key used to derive the `Rolling Proximity Identifier`) and also depends on the current `Rolling Proximity Identifier` [2].

3.7.2 Payload of the StopCovid application

The StopCovid[7] application deployed by the French government and based on the ROBERT protocol [8] uses the `Hybrid Model`. As the GAEN, StopCovid relies on a `Service Data` field associated to a dedicated service (UUID `0xFD64`) to carry the contact tracing data. This contact tracing data is composed of, a 1 byte encrypted country code (ECC), an `EBID` (the 8 bytes temporary identifier of the ROBERT protocol), a timestamp (epoch encoding on 2 bytes), a message authentication code (HMAC of EBID and timestamp) and metadata including version, transmission power level (for RSSI correction [14]) and two reserved bytes.

Overall the advertising payload is presented in Figure 7 and is organized as follows:

- `Flags` (3 bytes): Not specific to exposure notification, but must be included in every `advertising packet`.
- `Complete 16-bit Service UUID` (4 bytes): carry the UUID of the dedicated risk notification service (`0xFD64`).
- `Service Data - 16-bit UUID` (24 bytes): carry the data associated to the risk notification service along with the risk notification UUID
  - UUID of the dedicated risk notification service (`0xFD64`).
  - `ECC` (1 byte): encrypted country code.
  - `EBID` (8 bytes): the temporary identifier used by the ROBERT protocol.
  - `Time` (2 bytes): encodes the current epoch.
  - `MAC` (5 bytes): message authentication code computed from `EBID` and `Time` using HMAC-SHA256.
  - Metadata (4 bytes): version on 1 byte, `txPowerLevel` on 1 byte, and 2 additional bytes reserved for future use.

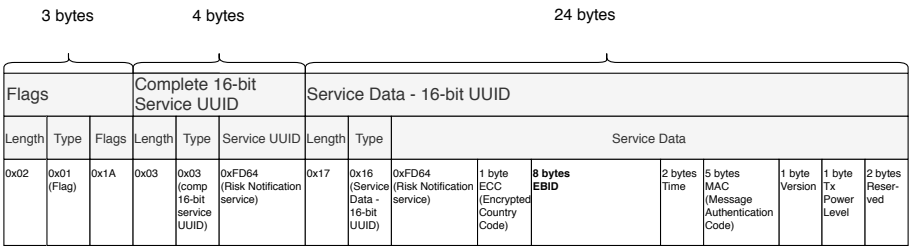

Figure 7: Payload of the advertising packet used by the StopCovid application. The `EBID`, timestamp, MAC and the metadata are carried by a `Service Data` structure associated to a risk notification service (UUID `0xFD64`)

4 OS functionalities and limitations

BLE is a feature included in all modern smartphones, nevertheless the usage of this wireless interface may be constrained by the operating system. In particular, all the functionalities are not directly available to an application, and when they exist, their usage may be subject to further limitation depending on the status of the application.

[7] `https://gitlab.inria.fr/stopcovid19/accueil`

4.1 Available BLE functionnalities

Operating systems offer access to BLE functionalities via an API. On Android an app has access to many BLE advertising functionalities: an app can start BLE advertising[8] and configure the advertising payload[9], and it can also scan advertising channels[10] and get full access to the payload of received advertising packets[11].

On iOS, the BLE advertising functionalities offered to an app are limited. An app has not a full control of the advertising payload. However, an app can partially control the content of advertising packets; for instance it can advertise a service UUID by advertising the service [12] or spawning an iBeacon[13]. iOS allows an application to perform scanning[14] and even though it does not provide full access to the advertising payload, it allows the app to retrieve data associated with the main element carrying data[15]. This is one of reason why the `Hybrid Model` has been considered.

ATT protocol functionalities required for the connected mode are available to applications on both Android and iOS. An app can connect to a peripheral on Android[16] and iOS[17]. Similarly, exposing services to allow incoming connections is possible on both Android[18] and iOS[19].

4.2 Background limitations

Functionalities offered by the OS might be limited by the status of the application. If the previously listed functionalities can be used by an app running in foreground, the access to those functionalities can be removed or limited when the app is in background.

For instance, on iOS the BLE scan interval as well as the advertising interval may be increased while in background[20]. An app running in background and advertising a service (for the connected mode) can only be discovered by other iOS devices.

Furthermore, experimental evaluations have shown that the behavior of background apps also depends on the status of the phone (sleep mode, screen off, ...) as well as on the phone model.

A possible solution to overcome those background limitations is to enforce that it remains in foreground (e.g. as a music player).

4.3 The Apple & Google Exposure Notification framework - GAEN

The GAEN framework, introduced by Apple and Google [1], offers to application, vetted by the two companies, functionalities to implement contact tracing.

An advantage of GAEN is that all the BLE operations are handled by the operating system. In particular, the GAEN framework take care of scanning the BLE channels and recording

[8] https://developer.android.com/reference/android/bluetooth/le/BluetoothLeAdvertiser
[9] https://developer.android.com/reference/android/bluetooth/le/AdvertiseData.Builder
[10] https://developer.android.com/reference/android/bluetooth/le/BluetoothLeScanner
[11] https://developer.android.com/reference/android/bluetooth/le/ScanResult, https://developer.android.com/reference/android/bluetooth/le/ScanRecord
[12] https://developer.apple.com/documentation/corebluetooth/cbperipheralmanager/1393252-startadvertising
[13] https://developer.apple.com/documentation/corelocation/turning_an_ios_device_into_an_ibeacon_device
[14] https://developer.apple.com/documentation/corebluetooth/cbcentralmanagerdelegate/1518937-centralmanager?language=objc
[15] https://developer.apple.com/documentation/corebluetooth/cbcentralmanagerdelegate/advertisement_data_retrieval_keys?language=objc
[16] https://developer.android.com/reference/android/bluetooth/BluetoothGatt
[17] https://developer.apple.com/documentation/corebluetooth/cbcentralmanager/1518766-connect
[18]
[19] https://developer.apple.com/documentation/corebluetooth/cbperipheralmanager
[20] https://developer.apple.com/library/archive/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/CoreBluetoothBackgroundProcessingForIOSApps/PerformingTasksWhileYourAppIsInTheBackground.html

identifiers, and also handles the broadcast and periodic rotation of identifiers. Since those operations are not managed by the application, background status is not a limitation anymore. Similarly, it is not necessary for the app to have a full access to advertising payload.

In addition to those benefits regarding BLE communications, the GAEN framework enforce constraints on the contact tracing system, in particular on the flow of information. As a consequence, only application following an approach of DP3-T can use the GAEN framework.

5 Overview of existing implementations

Several BLE-based contact tracing applications have been created in reaction to the COVID pandemic. Some of those applications have been developed by country while others have been developed by researchers and technical consortiums. In table 1 we present a list of identified BLE contact tracing applications[21].

Diverse technical choices have been made in those applications. Out of the 11 considered applications, 5 follow the `Broadcast Model`, 4 the `Connected Model`, and 3 the `Hybrid Model`. Furthermore, we can note that 4 of them are using the GAEN. Most applications support the two main platforms iOS and Android.

Table 1: List of BLE-based contact tracing application, with communication model, country.

System	Country	Model	Android	iOS	Comment
TraceTogether	Singapore	`Connected`	✓	✓	Based on BlueTrace [3]
SwissCovid [22]	Swiss	`Broadcast`	✓	✓	Apple-Google Framework
DP3T v1.0[18][23]	-	`Broadcast`	✓	✓	
StopCovid[24] [8]	France	`Hybrid`	✓	✓	
CovidWatch[25]	-	`Hybrid`	✓	✓	
NHS Covid App[26]	UK	`Connected`	✓	✓	
Covid Safe[27]	Australia	`Connected`	✓	✓	Based on BlueTrace [3]
Stopp Corona[28]	Austria	`Broadcast`	✓	✓	Apple-Google Framework
eRouska[29]	Czech	`Hybrid`	✓	✓	
Aarogya Setu[30]	India	`Connected`	✓		
Apturi Covid	Latvia	`Broadcast`	✓	✓	Apple-Google Framework
Corona-Warn-App [31]	Germany	`Broadcast`	✓	✓	Apple-Google Framework

6 Extension of the broadcast model

Some contact tracing systems, may require the transmission of contact tracing data larger than what can be carried by an advertising packet (28 bytes). We present here a number of solution that can be used to increase the amount of data transmitted that can be transmitted with the broadcast model.

An overview of those extensions along with pros and cons is presented in Table 2.

6.1 Scan response

The BLE [6, Vol 6, Part B, sec. 4.4.2.3] features a scan response that can be leveraged to transmit additional information between an advertiser and a scanner [7]. When receiving an advertisement packet, a scanner can send a scan request to the advertiser to which the latter

[21] This list is based on the contact-tracing app inventory of Technology Review `https://www.technologyreview.com/2020/05/07/1000961/launching-mittr-covid-tracing-tracker/`

Table 2: Pros and Cons of BLE broadcast model extensions.

Extension	Pros	Cons
Scan response	- Adds 31 bytes of adv. data	- Not broadcast: an exchange is required for each device
Extended advertising	- Adds up to 254 bytes of adv. data	- Only supported by latest OS versions
Rotating payload (Carousel)	- Size of adv. data increased by a factor k	- Contact detection time linearly depends on k - Rotation frequency may be limited in background
Multi advertising	- Size of adv. data increased by a factor k	- Only supported on Android 5+ - Not supported by all chipsets

should answer with a scan response (see Figure 8). A scan response has a payload of 31 bytes and could thus increase the amount of transmitted data to 59 bytes.

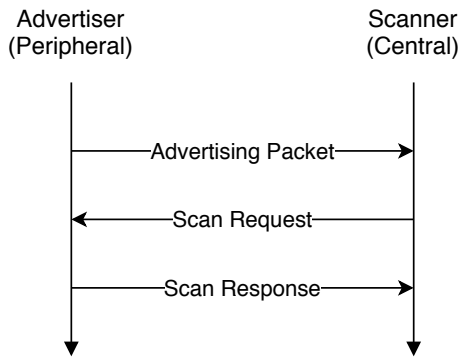

Figure 8: Scan request/response mechanism of BLE advertising.

One drawback of this approach is that the scan request and scan response packets are directed to a specific device, meaning that the information they carry is not broacasted and will be only received by one device.

6.2 Extended advertising

The v5.0 of Bluetooth introduced the *Extended Advertising* feature, that allows advertising up to 254 bytes [6, Vol 6, Part B, sec. 2.3.4]. This extra space can be used to transmit larger contact tracing data.

This extended advertising feature is supported by latest Android versions[32] and partially under iOS (only support extended advertisement payload up to 124 bytes)[33]

6.3 Rotating advertising payload (Carousel)

Increasing the amount of transmitted data could be done by rotating the advertising payload periodically. With this solution, the contact tracing data is split into several blocks and each block is advertised one after the other in a carousel fashion [16] (see Figure 9). To retrieve the complete contact tracing data, it is necessary to receive all the blocks. The rotation frequency

[32] `https://source.android.com/devices/bluetooth/ble_advertising`, see the method `isLeExtendedAdvertisingSupported()`

[33] `https://developer.apple.com/videos/play/wwdc2019/901/`

will determine the duration necessary to receive all the blocks. This block rotation is triggered by the app. This rotation frequency can thus be limited by the background constraints of the OS.

A Forward Error Correction (FEC) coding [15] could be added to increase the efficiency of this scheme, and of any scheme splitting contact tracing data over multiple advertising packets.

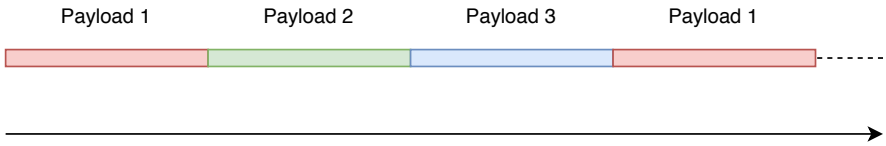

Figure 9: Carousel approach for advertising multiple advertising payload. Payloads are transmitted in sequential order, one after the other.

6.4 Multi-advertising

Simultaneously advertising multiple advertising payloads is a solution to increase the amount of data that can be transmitted. Using this approach multiple different advertising payloads can be advertised in parallel (see Figure 10). This feature, called *Multi-advertising*, is available on Android 5+ and allow to have multiple advertisement train running in parallel[34] and thus broadcasting an increased amount of data. This feature is not part of the Bluetooth specification, but can be seen as a device emulating several *peripheral* at the same time. Note that the support of Multi-advertising may depend on the Bluetooth chipset of the smartphone[35].

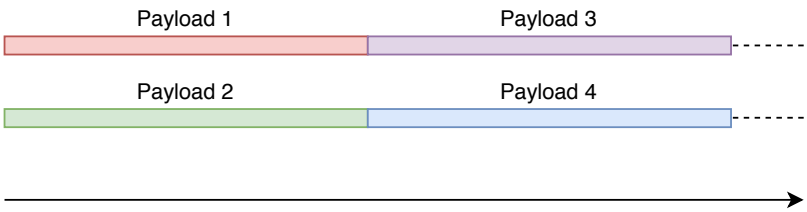

Figure 10: Multi-advertising approach for advertising multiple advertising payload. Multiple payloads are advertised in parallel.

7 Conclusion

Contact tracing can be implemented via BLE either following a `Broadcast Model`, a `Connected Model`, or a mix of them. BLE is supported on the two main mobile platforms Android and iOS, but the OS may restrain the use of BLE functionalities. The contact tracing introduced by Apple and Google can remove those constraints.

Size of advertising packets currently limits contact tracing systems based on the broadcast model to 25 bytes of useful data; however advanced BLE features offered by operating systems and the most recent Bluetooth specification could enable the transmission of larger amounts of data, opening the way for enhanced contact tracing protocols.

[34] `https://stackoverflow.com/questions/35033013/what-is-bluetooth-le-multi-advertising`
[35] `https://stackoverflow.com/questions/26482611/chipsets-devices-supporting-android-5-ble-peripheral-mode`

References

[1] Apple / Google. Exposure Notification - Bluetooth Specification v1.2. Technical report, April 2020.

[2] Apple / Google. Exposure Notification - Cryptography Specification v1.2. Technical report, April 2020.

[3] Jason Bay, Joel Kek, Alvin Tan, Chai Sheng Hau, Lai Yongquan, Janice Tan, and Tang Anh Quy. BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders. Technical report, November 2019.

[4] Johannes K Becker, David Li, and David Starobinski. Tracking Anonymized Bluetooth Devices. *Proceedings on Privacy Enhancing Technologies*, 2019(3):50–65, July 2019.

[5] Bluetooth SIG. *Bluetooth Core Specification v4.0*. June 2010.

[6] Bluetooth SIG. *Bluetooth Core Specification v5.2*. December 2019.

[7] Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. DESIRE: A Third Way for a European Exposure Notification System Leveraging the best of centralized and decentralized systems. working paper or preprint, May 2020.

[8] Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. ROBERT: ROBust and privacy-presERving proximity Tracing. Technical report, May 2020.

[9] Guillaume Celosia and Mathieu Cunche. Saving Private Addresses: An Analysis of Privacy Issues in the Bluetooth-Low-Energy Advertising Mechanism. In *MobiQuitous 2019 - 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 1–10, Houston, United States, December 2019. Core A, 28.67%.

[10] Guillaume Celosia and Mathieu Cunche. Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols. *Proceedings on Privacy Enhancing Technologies*, 2020(1):26–46, January 2020.

[11] Ranveer Chandra, Jitendra Padhye, Lenin Ravindranath, and Alec Wolman. Beacon-stuffing: Wi-fi without associations. In *Eighth IEEE Workshop on Mobile Computing Systems and Applications*, pages 53–57. IEEE, 2007.

[12] DP3T consortium. DP-3T/dp3t-sdk-ios, May 2020. original-date: 2020-04-13T15:17:06Z.

[13] Jean-Marie Gorce, Malcolm Egan, and Rémi Gribonval. An efficient algorithm to estimate Covid-19 infectiousness risk from BLE-RSSI measurements. Research Report RR-9345, Inria Grenoble Rhône-Alpes, May 2020.

[14] Jean-Marie Gorce, Malcolm Egan, and Rémi Gribonval. An efficient algorithm to estimate Covid-19 infectiousness risk from BLE-RSSI measurements. Research Report RR-9345, Inria Grenoble Rhône-Alpes, May 2020.

[15] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo. Reed-solomon forward error correction (fec) schemes. RFC 5510, RFC Editor, April 2009.

[16] Christoph Neumann and Vincent Roca. Analysis of fec codes for partially reliable media broadcasting schemes. In *International Workshop on Multimedia Interactive Protocols and Systems*, pages 108–119. Springer, 2004.

[17] TCNCoalition/TCN. Specification and reference implementation of the TCN Protocol for decentralized, privacy-preserving contact tracing. Technical report, TCN Coalition, May 2020. original-date: 2020-03-31.

[18] Carmela Troncoso, Mathias Payer, Marcel Salathé, James Larus, Dr Wouter Lueks, Theresa Stadler, Dr Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth Paterson, Srdjan Capkun, David Basin, Dennis Jackson, KU Leuven, Bart Preneel, Nigel Smart, Dr Dave Singelee, Dr Aysajan Abidin, TU Delft, Seda Guerses, and Cas Cremers. Decentralized Privacy-Preserving Proximity Tracing. Technical Report v1.0, April 2020.