Interoperable Digital Proximity Tracing Protocol (IDPT)

Jorge García-Vidal, UPC, BSC-CNS

Version 1, 18/05/2020

This is a first version of a document that describes the IDPT protocol. We are working on another document in which we make an analysis of privacy, security and implementation of this protocol.

Introduction

This document introduces the Interoperable Digital Proximity Tracing (IDPT) protocol, that can be run by applications that also run the DP3T digital proximity tracking protocol[1] to enable interoperability with the ROBERT digital proximity tracking protocol[2]. We believe that the same mechanism can be adapted to allow interoperability between other decentralized and centralized digital proximity tracing protocols, but analysis of this is kept out of the scope of the document.

The IDPT protocol avoids the reidentification attack of positive-tested users of the centralised system that was claimed to be an inherent property of interoperability systems[3], as in IDPT the system **does not publish the list of decentralised ephemeral identifier that were at risk of exposure of users of app R**.

As is well known, the current iOS and Android Exposure Notification API[4] only supports protocols of the distributed class. Due to this lack of support for centralized approaches, the implementation of the ICDT protocol has the same known difficulties as ROBERT, which appear mainly when applications are running in the background. Additionally, since devices must transmit two types of BLE beacons, we expect that devices running ICDT will have higher power consumption compared to implementing a pure DP3T mechanism.

We believe that, in practice, the use of ICDT in countries where the majority of users use DP3T-type applications should be optional, mainly in areas where the presence of R nodes is likely (for example, large cities, tourist areas, airports, etc.). Another situation in which the use of ISPT could be adequate is when a user of the app I visits a country where the majority of the population uses R application. In addition, a country could introduce a DP3T application in a first phase, and only later incorporate ICDT.

Assumptions and notation

[1] https://github.com/DP-3T/documents/blob/master/DP3T%20White%20Paper.pdf
[2] https://github.com/ROBERT-proximity-tracing/documents/blob/master/ROBERT-specification-EN-v1_0.pdf
[3] U. Lukas et al. "Interoperability of decentralized proximity tracing systems across regions"
[4] https://developer.apple.com/documentation/exposurenotification

We assume that in the same geographic area we have users of 3 different types of digital proximity tracking applications: R, D and I:

- Application R implements the ROBERT protocol.
- Application D implements the DP3T protocol
- Application I implements the DP3T + ICDT protocols, that is, DP3T with the added ICDT protocol.

We assume that applications I and D interoperate, which means that if a user of the application D/I reports to the app a positive tests, devices of users of the application I/D who were exposed will be notified

In addition, we will see that we can have interoperability between the I and R applications, meaning that **if a user of the application R/I reports to the app a positive test, devices of users of the application I/R who were exposed will be also notified**.

As we will continuously refer to devices, users, and backend servers from the three protocols, we introduce the following notation to simplify the description of the mechanism:

- For X = R, D and I, "X-device" is a mobile device that runs an instance of a X-type application, "X backend server" is the backend server used by an application X, while “X-user” refers to a user of an application X.
- As prescribed by ROBERT, the R backend server generates ECC+EBID values, which will be called R-ECC+EBID. Additionally, when these fields are transmitted in a beacon, the fields “Time” and “MAC” are added,
- As prescribed by DP3T, nodes D and I generate EphID values, called D-EphID and I-EphID respectively.
- I-devices also generate EBID values called I-EBID. I-EBIDs have not attached an ECC field.
- ICDP requires the use of a relay, which we call I-relay.

The R-ECC+EBID+Time+MAC, D-EphID, I-EphID and I-EBID values are 16-byte strings, transmitted in the payload of packets type ADV_IND Bluetooth Low Energy, which we will call BLE-beacons or just beacons.

We assume that there are some metadata that distinguish the beacons emitted by different types of applications. More specifically, in the case of the app I, we assume that it is possible to distinguish beacons carrying I-EphIDs and I-EBIDs

Backend server and relay

Protocols DP3T and ROBERT require the use of back-end servers.

The main functionalities of ROBERT backend servers are:

- Generate Ephemeral Bluetooth IDs (EBID) and Encrypted Country Codes (ECC) for each R-device. EBID = ENC(K_s, epoch; ID), where K_s is a cryptographic key on the R backend server server, while ECC = MSB(AES(K_G, EBID | 0^{64})) XOR CCA, where CCA is a country code, and K_G is a key shared among the federated backend servers. The epoch is a discretization of time in units of 15 minutes. R-devices regularly pull these values.
- Maintain an IDTable that allows the back-end server and uses a ROBERT risk score algorithm for determine the exposed users of the application R, this being

one of the main differences with a protocol such as DP3T, in which the calculation of the risk score is performed in the mobile device.
- Be responsible for federation with other application servers that interoperate with application R.

The main functionalities of DP3T backend servers and I backend servers are:

- Transmit a compacted representation of the EphIDs of the users who reported a positive test to the devices of other users of the application. Using this information, mobile devices can verify if they have received these EphIDs, which means that they were in proximity of a user who tested positive. A risk score is evaluated in the devices -this is a main feature of DP3T- using parameters such as the intensity of the received Bluetooth signal or the duration of the contact to quantify the risk of exposure to users.
- Achieve interoperability of DP3T applications.

IDPT requires additionally the use of a relay node, that we call I-relay. Its main functionalities are:

- Allow the exchange with the R backend server of ephemeral identifiers of devices of R-users and I-users applications who tested positive.
- Post lists of values that allow I-devices check whether they were in exposure contact with a device of a user of the application R who reported a positive test.

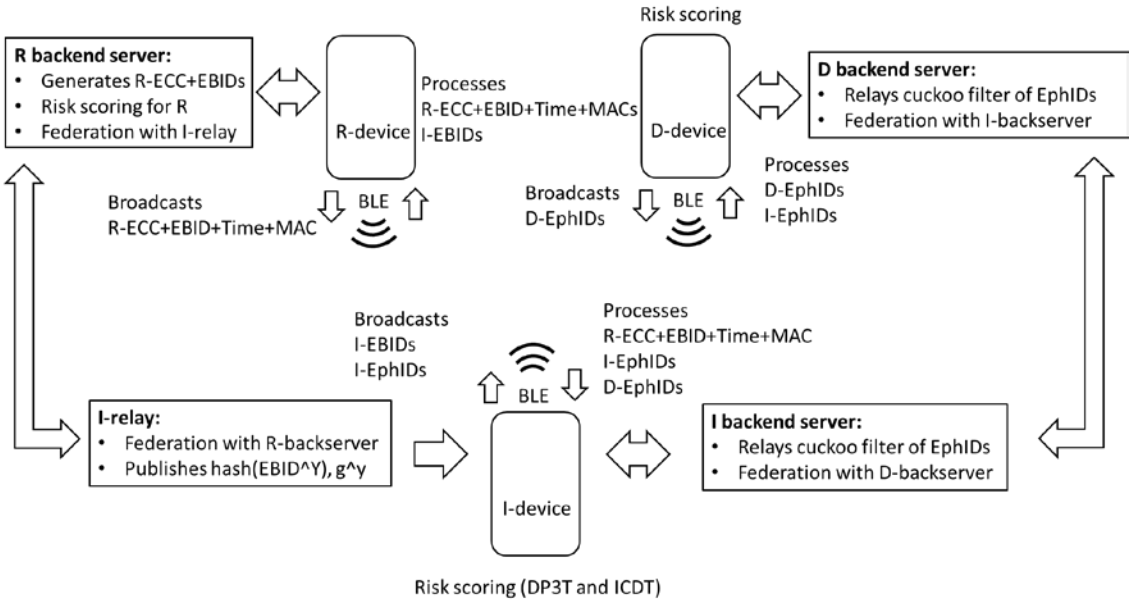

Figure 1: Functional elements

Ephemeral IDs generation

The R backend server is responsible of generating the R-ECC-EBIDs. In DP3T, however, the devices are the ones responsible of generating the EphIDs, using a hash of a secret seed, which is changed every time epoch.

Also, and this is a key feature of IDPT, **I apps generate another kind of ephemeral bluetooth ID that we will call I-EBID**. I-EBIDs are calculated as in a Diffie-Hellman exchange: each I-device generates a sequence of secret numbers $\{X_n\}$ and computes a sequence $\{\text{I-EBID}_n\} = g^{X_n}$. These X_n values are stored in the I-device for several days

(for example, 21 days) and are kept secret. The frequency at which these X_n values are generated is discussed in the section “Risk Scoring for ICDT”.

The format of the 3 types of payloads transmitted in the BLE beacons by the apps R, I and D is shown in the figure:

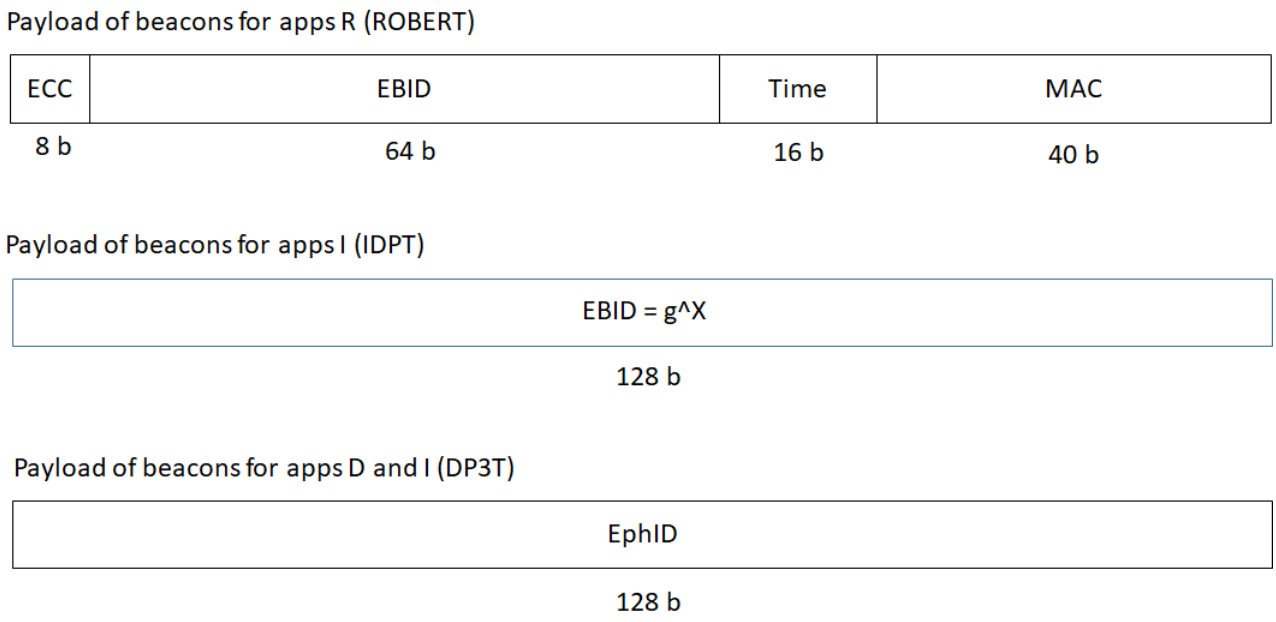

Figure 2: 16 byte payloads of ADV_IND BLE packets

Note that ROBERT includes some fields in the beacon payloads (Time and MAC) that prevents against identity attacks and mitigates against replay attacks.

Federation and back-end server interconnection

The federation between applications that use ROBERT is based on the use of the field ECC, that encrypts a country code. This country codes identifies whether the data (R-ECC+EBID+Time+MAC, time) must be processed locally or forwarded to another ROBERT backend. For achieving interoperability with IDPT, the R backend server must be able to identify the data to be forwarded to the I-relay by another means, as I-EBIDs are transmitted without an encrypted country code.

Proximity Discovery

- R-devices broadcast R-ECC + EBIDs plus two fields “Time” and “MAC” on the payload of Bluetooth Low Energy (BLE) ADV packets (BLE beacons), which have a payload of 16 bytes.
- D-devices and I-devices transmit D-EphID and I-EphID on BLE beacons.
- I-devices also broadcast I-EBID on BLE beacons.

Ephemeral IDs processing

- R-devices store R-ECC+EBID+Time+MAC and I-EBIDs.
- D-devices store D-EphIDs and I-EphIDs

- I-devices store I-EphIDs, D-EphIDs and R- ECC+EBID+Time+MACs.

This internal storage add time of reception and possible additional information, such as the strength of the BLE signal.

Exposure Status notifications

For all three types of applications, when a user tests positive, she receives an authentication token and voluntarily decides if she will use this token to report the application of the positive test. If the user decides to report the application, the following procedures are followed.

User of app D reports a positive test

Application D reacts as prescribed by the DP3T protocol, and since we assume interoperability between applications D and I, devices of exposed users of applications I and D are notified. The I-devices and D-devices applies a DP3T Risk Scoring algorithm to decide whether users should be notified. Unfortunately, R-devices applications do not receive notifications, since R and D do not interact.

User of app I reports a positive test

The application reacts as prescribed by the DP3T protocol, which means that devices of users of both I and D apps are notified as we are assuming interoperability between the two applications. Again, devices apply a Risk Scoring algorithm to decide whether users should be notified.

In addition, the application transfers the received R-ECC+EBID+Time+MAC, which have been stored for several days (for example, 21 days), to the I-relay. I-devices do not transfer received I-EBIDS to the I-relay, as received I-EBIDs are not even stored by I-devices. This is not necessary since the other I-users are notified using the distributed mechanisms.

The I-relay de-encrypts the field ECC from the received R-ECC+EBIDs and forwards the information to the R backend server[5]. The R backend server runs a Risk Scoring algorithm which decides which users of apps R who were in contact with an I-user who tested positive are notified.

In other words, if an I-user is positive, exposed devices of applications I, D, and R are notified.

Obviously, the I-relay cannot de-encrypt the EBID field since these EBIDs were generated by R backend server using a secret key K_S. The I-relay cannot identify R-users or I-users, does not keep any permanent table, and cannot create a contact graph.

User of app R reports a positive test

The application reacts as prescribed by the ROBERT protocol: it transfers the received R-ECC+EBID+Time+MAC and I-EBIDs (i.e. the *LocalProximityList*) to the R backend server. This LocalProximityList must include now a field that allows the identification of I-EBIDs for a given I application.

[5] In this document we do not address the case of multiple R applications in the same geographical area

The R backend server will separate the I-EBIDs from the R-ECC+EBID+Time+MACs. The R-ECC+EBID+Time+MACs are processed locally by the R backend server, which runs a risk scoring algorithm that decides which R-users should be notified.

The I-relay receives the I-EBID list, and generates one secret numbers Y per received I-EBID[6]. For each I-EBID, it picks one of these numbers Y, and computes (I-EBID)^Y. This value will be equal to g^XY for some secret value X of an I-user who was in contact with the R-user who reported a positive test.

The I-relay publishes a list of values {hash(g^XY)} and values {g^Y}.

I-devices periodically pull this list, e.g. twice daily. They compute hash((g^Y)^X_n) = hash(g^X_nY) for the values stored in the sequence {X_n} that were generated by the device and all the values {g^Y}. Then, it checks if there is a match of the calculated values and some of the hashes in the list. It also determines the time epoch of the exposition.

Once the device determines the matches, the application can run a risk scoring algorithm that decides which R-users are notified. This algorithm is discussed in the next section.

As a conclusion, we see that if an R-user is positive, exposed devices of applications R, and I are notified. Again, D-devices are not notified as R and D do not interoperate.

Risk Scoring for IDPT

The algorithm that evaluates the risk scoring from exposures with R-users who have reported a positive test would be different from the one used for DP3T. The main difference comes from the fact that in general the device cannot determine the R-ECC+EBID+Time+MACs of the users who reported a positive test. The only information it has is the set of received R-ECC+EBIDs received during the time interval during which the X_n that indicated an exposure were transmitted.

In order to reduce the uncertainty, one option is to increase the frequency at which the values X_n are generated (e.g. generate a different X_n per minute), thus increasing the time resolution. There is a trade-off, however, between this frequency and the amount of computations that a I-device must perform in order to assess its exposure, that will be proportional to this rate.

What should be modified?

- The system D does not need any change with regards to the standard decentralized approach.
- The R apps should be modified in order to recognize the beacons with I-EBIDs generated by I devices. The *LocalProximityList* uploaded to the R backend server in case the user reports a positive test should be modified to indicate the two different types of EBIDs (ECC+EBID+Time+MACs and I-EBIDs) to the R backend server.
- The R backend server must be able to recognise the I-EBIDs and transfer these values to the I-relay,
- In the I-devices, we must generate two types of content for the BLE beacons (i.e. I-EphIDs and I-EBIDs).

[6] Alternatively, the I-relay could chose a limited number of Ys, and re-use the same Y to compute EBID^Y, thus reducing the computations at the I-devices.

- The I app must have a faster beacon rate, that could lead to higher power consumption.
- The ICDT implementation will not be supported by the current Exposure Notification API of Google and Apple, meaning that it will suffer from the same implementation difficulties of other centralized protocols as ROBERT.
- The BLE beacons must have some metadata that makes it possible for another application that receives the beacons to differentiate the 4 types of content: ECC+EBID+Time+MACs, D-EphID, I-EphID, I-EBID.
- We need to add an I-relay

Acknowledgements

I would like to acknowledge the useful comments of Leonardo Bautista Gómez, Pablo Rodriguez Rodriguez, Pedro Martín Jurado and Carlos Pastor. However, any errors in the text are the sole responsibility of the author.