# IDPT-FP (IDPT, Full Protocol)

Jorge García Vidal, UPC, BSC

Draft version May 23th 2020

This is a proposal for a digital proximity tracing protocol that can operate both in centralized and distributed mode with full interoperability. It is based on the mechanism described for the IDPT protocol[1] for interoperability between ROBERT[2] and DP3T[3] applications.

## Assumptions

We assume that in the same geographical area (e.g. the Schengen area) we have a digital proximity tracking application with users who can decide whether the risk score is done on a central server (users C), or is done on the user's phones (users D).

It may be that in a country within this geographical area, national public health authorities choose to support only users C or users D. Another option is that they give freedom of choice to their citizens, who assess the trade-offs between privacy and security and the effectiveness of the risk score.
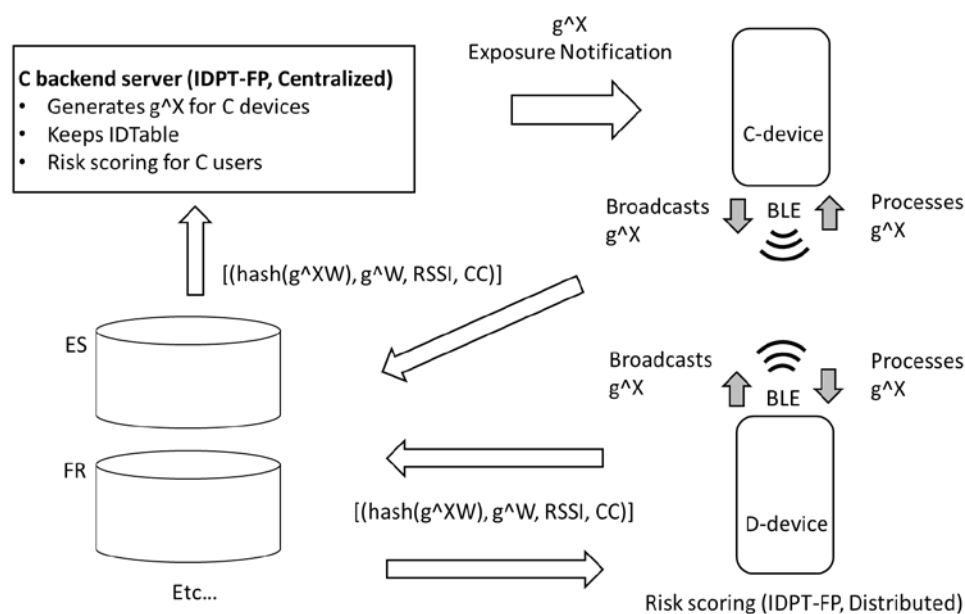


**Figure: Functional elements**

[1] https://github.com/IDPTdocs/documents/blob/master/IDPT-v2.pdf

[2] https://github.com/ROBERT-proximity-tracing/documents/blob/master/ROBERT-specification-EN-v1_0.pdf

[3] https://github.com/DP-3T/documents/blob/master/DP3T%20White%20Paper.pdf

# Beacon broadcast

- All users broadcast BLE ADV_IND packets (which we call "beacons") with a payload equal to g^X, where X is a secret number that is changed at each epoch (e.g. 15 minutes) .
- We assume that g^X is a 16 byte number.

# g^X generation

- For C users, g^X is generated in a central server (C-backend server), which knows the sequence of secret values X. The C-backend server maintains a IDTable associated with these secret values (this can be X=hash(ID,epoch), etc), with a structure similar to the one used in ROBERT.
- For D users, g^X is generated in the devices, which keep the sequence of secret values X

# Beacon processing

All users retain the g^X values received, as well as the RSSI of the received beacon.

# Users with tests COVID+

### User C is tested COVID+

Devices C choose a secret random number W (which is not known to server C), and insert the list of tuples [(hash(g^XW), g^W, RSSI)] into a global public list, so that the identity of the device remains anonymous.

### User D is tested COVID+

Devices D choose a secret random number W, different according to the received g^X, and insert the list of tuples [(hash(g^XW), g^W, RSSI)] into a global public list, so that the identity of the device remains anonymous.

The tuples are kept in the public list for a limited period of time (e.g. 1 day). The global list can be organized into geographic areas, in order to reduce the download traffic for D-devices; see section "Support for Roaming".

# Exposure notification

### User C

The C-backend server periodically reads the public list [(hash(g^XW), g^W, RSSI)]. Then it looks for intersections of hash((g^W)^X), for the X values stored in the IDTable, with the hash values of the list [(hash(g^XW), g^W, RSSI)].

For C users, who use the same W value for all published tuples in the list, the IDTable can obtain a time series of contacts for the risk scoring algorithm (because the user with test COVID+ publishes a constant g^W value). If the user who tested COVID+ is a user D, the time series information is lost.

Note, however, that the C-backend server does not know the identity of the users (C or D) who tested COVID+, because the W value is kept secret in the device.

The C-devices periodically connect to the C-backend server for receiving an Exposure Notification.

**Users D**

D devices periodically read the public list [(hash($g$^XW), $g$^W, RSSI)]. Then, they look for the intersections of the hash(($g$^W)^X) ), for the X values stored in the device, with the hash values in the list [(hash($g$^XW), $g$^W, RSSI)].

A Risk Scoring is then run in the D-device that decides whether the D user is notified as in risk of exposure.

We discuss later how to reduce the number of computations in the case of D users by using Country Codes.

# Risk scoring

The proximity of the contact can be estimated from the RSSI value. The length of the contact is obtained from the times when the X value was used. Centralized backend servers can use time series in their risk scoring algorithms, since the W value in the C devices is the same for all declared tuples (hash($g$^XW), $g$^W, RSSI).

# Support to roaming

Devices could add a country code (e.g. CC = "ES", "FR", etc.) to each tuple in the published list: [(hash($g$^XW), $g$^W, RSSI, CC)].

This country code corresponds to the place where the $g$^X was received. Users should specify the country where they are located to the application. If no location is given, a special code can be used ("Schengen"). Since the identity of users is kept anonymous, users do not reveal publicly their location.

The C backend server knows the location where the interaction of the user who transmitted $g$^X took place, but this is already the situation in ROBERT in the case of federated backend servers.

Note that users D have the advanbeacone now that their location remains private.

**C-Users**

C-backend servers must check the intersections for the entire list published in the Schengen area. This calculation is indeed long (14*100*length([(hash($g$^XW), $g$^W, RSSI, CC)]) but it is performed twice a day, and should not be a major problem for a backend server.

**D-Users**

Device D should only evaluate the hash(($g$^W)^X) for elements with a CC that corresponds to one of the zones visited in the last 14 days.

# Defences again other attacks

The device can add to the list two more fields: hash(g^X + epoch), hash(g^X + MAC)[4], where epoch is the time of reception of g^X, and MAC is the MAC address of the beacon that contained g^X. These fields need to be checked only in case of intersections.

# Consequences of breaking Diffie-Hellman

If an attacker breaks DH (i.e. gets W from g^W), she could check if a g^X that has been eavesdropped matches the hash(g^XW) value. This would lead to the conclusion that the user who transmitted g^X was close to a user who reported a COVID+ test. This vulnerability is inherent in all digital proximity tracing protocols, and the attacker can obtain this information in a much simpler way.

# Possible implementation issues

- Support of Gapple for this protocol.
- Is 16 byte Diffie-Hellman too weak?. We think that the information that an attacker can obtain from breaking DH is not worth the effort. However, if this is considered a risk, using 32 bytes DH and the corresponding consequences on beacon transmissions should be considered.

# Privacy properties

The D-users avoid re-identification attacks of distributed protocols such as DP3T. C-users share less information with the C-backend server, as users are keep anonymous when reporting a test COVID+.

# Acknowledgement

---

[4] Is this enough: hash(g^X + epoch + MAC) ?