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 (C-users), or is done on the user's phones (D-users).

It may be that in a country within this geographical area, national public health authorities choose to support only C-users or D-users. 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.

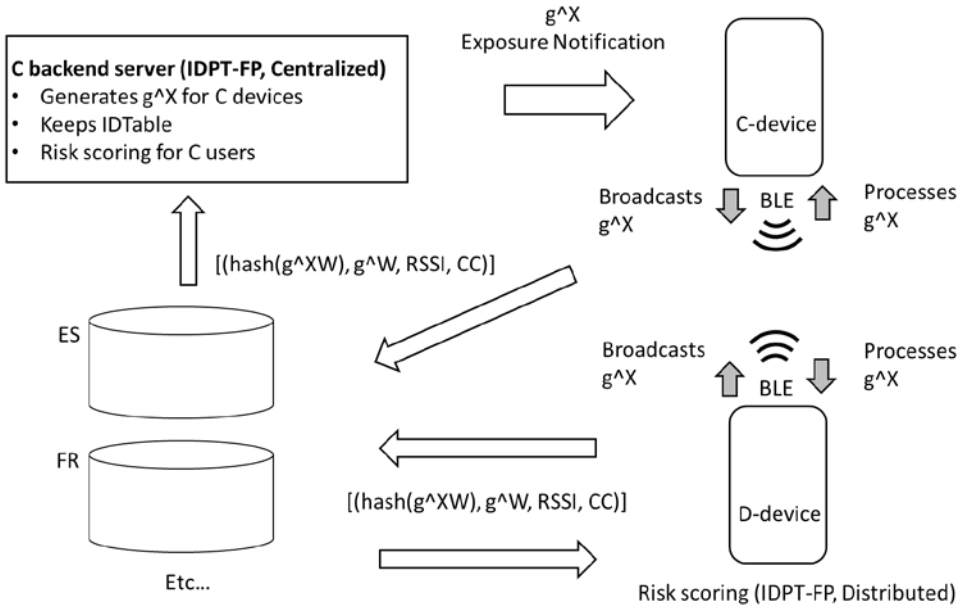

Figure: Functional elements

[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, as in a Diffie-Hellman exchange, 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 with a structure similar to the one used in ROBERT, with entries associated with these secret values. These values could be obtained, for instance, as X = hash(ID + epoch), where ID is an pseudonymous associated with an C-user.
- 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+

C-user is tested COVID+

C-devices 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.

D-user is tested COVID+

D-devices choose a secret random number W, different for each 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 for both C-users and D-users are kept in the same public list with a keep-alive time of 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

C-Users

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 D-user, 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.

D-Users

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 duration and date of the contact are obtained from the epochs when the matching X value was used. Centralized backend servers can use time series in their risk scoring algorithms when the user who tested COVID+ was a C-user, since the W value in the C-devices is the same for all declared tuples (hash(g^XW), g^W, RSSI).

Support to roaming

One solution is possible if applications can differentiate between beacons transmitted by application of different countries. In this case, for instance, the devices could add a country code (e.g. CC = "Spain", "France", etc.) to each tuple in the published list: [(hash(g^XW), g^W, RSSI, CC)]. CC could correspond to the country of the app of the transmitted beacon.

D-devices and C-backend servers must check the intersections for the entire list for tuples with CC equal to the corresponding application. For each D-device, this calculation requires (14*100*length([(hash(g^XW), g^W, RSSI, CC==App)]). For the C-backend server this calculation must be done per entry in the IDTable. This computation is performed, for instance, twice a day[4].

C-users would reveal to C-backend server no information about roaming. Note that, in the case of ROBERT, the backend server knows the federated server of the user who transmitted the ECC+EBID.

[4] If this number becomes too large, one can think on alternative solutions: e.g. computations done in the C-devices, which should know the values X. The C-devices would then report back their results to the C-backend server. In this case, the differences between the C-devices and D-devices would almost disappear.

For D-users, the visited countries remain private.

If this differentiation between apps is not possible, this country code CC could correspond to the place where the g^X was received. In this case, the 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.

In this second case, the C-backend server knows the location where the interaction of the user who transmitted g^X took place. For D-users, the visited countries would remain private. C-backend server should check for intersections for the whole list, whiele D-devices only for tuples with CC equal to one of the visited countries, as reported by the user to the application.

Defences again other attacks

Devices add to the list two more fields: hash(g^X + epoch), hash(g^X + MAC)[5], 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

- The current Gapple EN API would not support this mode of operation.
- Is 16-byte Diffie-Hellman too weak?. We do not think so, as the information that an attacker can obtain from breaking DH is not worth the effort. However, in case this is considered a risk, using 32 bytes DH and the corresponding consequences on beacon transmissions should be considered.

Privacy properties

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

The author would like to thank Leonardo Bautista-Gómez, Pedro Martín-Jurado, and Pablo Rodríguez-Rodríguez for useful discussions on the properties of this protocol. The fact that IDPT can be used as a full digital proximity tracing protocol was clear from the beginning, but the author made no effort to work out the details until Pablo pointed out that this protocol would probably not suffer from the re-identification problems of other

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

distributed protocols. However, any errors in the text are the sole responsibility of the author.