A simple proximity-based approach to contact tracing

By team-covid-19-contact-discovery project [1]
Version 3/23/2020

We describe here a simple proximity-based approach to using smartphones for contact tracing based on Bluetooth proximity (a plausible proxy for being within covid-19 transmission range).

This approach is essentially identical to that in the Covid-Watch proposal https://www.covid-watch.org/articles. We would be happy to see technical convergence on any of the details. (The biggest of which may be the possible utilization of the *FindMy* beacons.)

Seed generation

- Each phone has a unique random secret 256-bit seed SK, known only to that phone (and/or its owner and possibly other devices with the same owner).

ID generation

- The phone generates a sequence ID_t of pseudorandom IDs, one for each 15-minute epoch, using its secret seed SK.
- For example, we might have ID_t = PRF(SK, t) were PRF is a pseudorandom function and t is the time (measured to 15-minute precision only).
- A phone needs to be able to re-generate all of the IDs it has used during the last two weeks or so. It can do this by just recording all the IDs it has used, or by recomputing using the functional definition of ID_t as given for example in the last bullet.
- We assume that each ID is at least 10 bytes and at most 28 bytes long. A minimum of 10 bytes assures minimal collisions, and 28 bytes is the length of an Apple FindMy key. Software should be prepared to deal with IDs with lengths in this range.
- The IDs for a given phone should appear random; no one seeing an ID for a phone can predict what any other ID for that phone will be at a different time, and no one can tell if two IDs came from the same phone or not.

Beaconing

- The phone periodically (every minute) broadcasts its current ID_t using Bluetooth or some other convenient technology (e.g. ultrasound).

Listening/Logging

---

[1] Team members include: Ron Rivest, Danny Weitzner, Yael Kalai, Ramesh Raskar, Jon Gruber, Kevin Esvelt,Praneer Vepakomma, Adi Shamir, Nickolai Esvelt, Hal Abelson. All from MIT except Adi Shamir, who is from Weizmann.

- A nearby phone (say, owned by Bob) listens for such IDs, and stores any that it hears. If it hears one, it stores it as a *contact entry* in its local *contact log*.
- A contract entry may store other information, such as
    - time of day when contact occurred
    - number of times this ID has been heard
    - GPS location,
    - signal strength
    - Bob's own ID at the time of contact (only necessary if it can't be recomputed)
- Bob's phone doesn't store entries for longer than three weeks; once a contact entry is more than three weeks old it is deleted from Bob's phone.
- Note the difference with *FindMy*: a contact entry is only stored locally in the listener's phone, and there is no immediate communication with Apple or any other party.

Permission numbers

- Each testing authority or doctor is given, or has access to, a set of "permission numbers" that will be used to authorize uploads of contact logs.
- Each such permission number might, for example, be a random nine-digit number.
- Each such permission number is "use once"; different doctors and testing authorities receive different sets of permission numbers. For example, they might have a page of sticky labels with each label having one such permission number.
- (The permission number scheme, and the term "permission number," are the same as in the Covid Watch proposal.)

Infection tests and uploading

- If Bob is tested and found positive for coronavirus, he is given a previously unused permission number. (By, say, taking the next unused sticky label from the sheet.)
- Bob (or his proxy) enters the permission number into the app running on Bob's phone. This authorizes Bob's phone to upload Bob's contact log to the "master exposure database". The permission number is checked by the server providing the upload service.
- Bob's contact log is uploaded. No record is kept of Bob's identity by the server.

Searching and alerts

- If Alice wants to know if she might have been exposed, she checks the master exposure database (or the relevant shards of it) to see if any of the IDs her phone has used in the last two weeks appear.
- If one or more of her IDs appear in the master exposure database, she may have been exposed to the coronavirus by a person who has tested positive for coronavirus; she

should follow the guidance of local medical officials, say by self-quarantining and by practicing social distancing.

- **Pre-registration of ID hashes:** The above approach requires an active inquiry by Alice; it does not include a means for Alice to remain passive and to be notified when one of her IDs appear in the master exposure database. This functionality could be supported if Alice "registers" the hashes of her IDs. Hashes are used to prevent the registrar from filing false reports.

Databases

- If the master exposure database is small, Alice can run an inquiry by downloading the database in its entirety, and then searching the downloaded copy.
- If the database is a bit large for Alice to download, it might be downloaded by an organization that she trusts, who could then do the relevant searching for her.
- The database can get quite large, if many people test positive. For example, if
    - A contact entry takes 64 bytes to store
    - A typical phone contact log contains 1024 entries
    - There are 16M people who have tested positive.

  then the master database will take about 1TB to store!
- If the master database becomes too large, it can be partitioned into modest-sized "shards" according to crude geographical position. Each such shard might contain 1000 contact events. A client would need to download several of these shards for its search. (Although the client has roughly 1400 relevant IDs, the number of relevant shards is likely to be small.)

Integration with Apple *FindMy*

It may be challenging to get a new app widely enough distributed to help contact tracing significantly reduce the spread of infection.

One way of "getting started more quickly" may be to leverage functionalities that are already deployed in existing phones.

Such a functionality, for example, exists in the Apple *FindMy* functionality.

With *FindMy,* each phone periodically broadcasts a 28-byte public key. This key changes every 15 minutes to protect the privacy of the phone's owner. We don't need to follow the *FindMy* protocol further here.

A natural extension to the protocol described here is to use the *FindMy* beacons as if they were broadcasting IDs, and not public keys. (More simply, we just treat the PKs as if they were IDs,

and don’t encrypt anything with them.) This provides a broad base of already-functioning beacons that can be used in the current protocol for contact tracing.

Additional features of the *FindMy* protocol may also be used, such as encrypting the GPS location of the contact with the public key received. This protects the location of the contact from being known by anyone other than the contacted party. In this variant (which we support), Any 28-byte ID is treated as a public key, and the GPS location in the contact entry would be encrypted with that public key (in the same manner as the encryption in *FindMy*).

Conclusion

We have sketched an approach to tracing contacts using smartphones. It is technically simple, and appears to be nearly identical to that of the Covid-Watch proposal. The ability to integrate Apple *FindMy* beacons may be novel.

We would hope to see this implemented soon, by many parties!