CrowdNotifier

Decentralized privacy-preserving presence tracing

Version 8 October 2020

Wouter Lueks (EPFL), Seda Gürses (TU Delft), Michael Veale (UCL), Edouard Bugnion (EPFL), Marcel Salathé (EPFL), Carmela Troncoso (EPFL)

Contact the first author for the latest version

Executive Summary

We are publishing this document to inform the discussion revolving around the design and implementation of proximity tracing systems. We seek feedback from a broad audience on the high-level design, its security and privacy properties; so that missing requirements and protections can be identified and the overall system can be improved.

Contract tracing aims to slow down the spread of viruses (e.g., SARS-CoV-2) by identifying contacts of persons that have been infected with the virus. Traditionally this is a manual effort where contact tracers interview SARS-CoV-2-positive people to determine who their contacts are. Digital proximity tracing aims to augment this effort by rapidly notifying contacts that have been in close proximity to an infected person, even if the infected person could themselves not identify that contact.

Epidemiologists have pointed out the need for *presence tracing* to notify people that have been sharing a semi-public location with a SARS-CoV-2-positive person for a prolonged time. Such locations include restaurants, bars, classrooms, and places of worship; but also events such as concerts and meetings. In contrast, digital *proximity* tracing, such as *Exposure Notification*, does not identify such contacts unless the user was in close physical proximity to the SARS-CoV-2-positive person.

Existing presence tracing systems typically require a user to “check-in” to a location, such as using a hand-written or electronic form, or using a sensor on a device, such as a camera to scan a QR code.

In this document we present a simple presence tracing system, CrowdNotifier, that is ***easy to use***, will ***effectively notify*** any presence contacts of SARS-CoV-2-positive persons, while at the same time providing ***strong privacy and abuse-prevention properties***.

Motivations

Checks on power and population control Digital presence tracing systems facilitate the notification of individuals who meet epidemiologically-defined criteria (e.g., were present at a location within a certain time window compared to a potential index case). Existing presence tracing systems do so at the sole discretion of a central authority such as a public health agency. Such infrastructure can, however, be easily abused if a central authority utilises it for non-epidemiological purposes: With a registry of locations, and background knowledge (e.g., about the demographics or characteristics of visitors to those locations), segments of the population can be ordered to quarantine with no additional infrastructure or cost.

This contrasts with proximity tracing technologies such as *Exposure Notification*, where to trigger notifications with this effect would require the deployment of physical infrastructure (ie., Bluetooth emitters) which could not operate with the same certainty, costless scalability, or undetectability. When combined with legal obligations to quarantine and/or individuals' personal desire not to infect others, systems of presence tracing that operate at the discretion of a central authority could serve as a method of population control. It could, for example, be used as a tool for voter suppression, particularly as political actors often already hold detailed background knowledge as to the location and demographics of voters holding different electoral intentions. As a consequence, digital presence tracking systems should be designed to effectively meet genuine epidemiological goals while introducing safeguards against this potential abuse of power by central authorities.

Safeguards for privacy A second group of concerns around presence tracing relates to issues of individual privacy. Systems which leak individuals' location histories can be abused in a variety of ways depending on to whom the leak occurs. Leaks to central authorities may result in persecution; leaks to other users, such as perpetrators of intimate partner violence and abuse, may exacerbate coercive control. Digital presence tracing systems should be designed to remove or minimise these risks.

Concerns for the confidentiality of locations A third group of concerns relates to the confidentiality of locations, and societal harms that can result from rendering the existence of social gatherings, meetings and communities centrally legible and machine-readable. While many venues already volunteer or are legally compelled to be part of large and often publicly accessible databases (e.g., bars for licensing or mapping purposes), this is not the case for all epidemiologically relevant gatherings. For some, such as political or religious gatherings, a technology that demands they formally register their existence to an entity outside their community (such as a central authority) may itself pose a threat regardless of the individual privacy protections or safeguards on the power to issue notifications. While technological safeguards may help mitigate some concerns, at the heart of this issue is the definition (which may be in law) of what constitutes a relevant 'location' for epidemiological purposes. Choosing to broaden this beyond existing, self-published locations (e.g., most restaurants, bars, cafés) will be a

political choice that cannot be easily specified or limited by a technological protocol, and the breadth of the definition should be considered in the proportionality of introducing any presence tracing system.

Additionally, the system should aim to maintain confidentiality of which locations were visited by SARS-CoV-2-positive people to prevent stigmatization and shaming of owners/organisers and visitors.

Gracious dismantling A final concern relates to the sunset of the system. Because the system is introduced in a situation of emergency, it is important to ensure that, after this situation ends, the impact of the system is minimal. Ideally, control on the sunset should be on the users' side, without creating a dependency on authorities. Successful sunset of the system should not rely on authorities dismantling infrastructure (e.g., by stopping servers, or deleting databases).

Moreover, it is desirable that once the system is not necessary anymore, authorities dismantle all infrastructure supporting the system. In other words, none of the infrastructure created to support the system should prevail after system operation stops.

Terminology

For the purpose of presence tracing we are interested in crowds that gather at a certain location at a certain time. This could be a friday evening at a bar, or a Tuesday afternoon meeting in a community center, or even an exhibition fair spread out over several buildings. For readability, however, we use the term "location" to refer to locations at a specific time as well as specific events that happen within one or more locations.

Location: a semi-public location (for example, bar, restaurant, religious building, events venue, or meeting place) or an event which takes place in one or more locations (for example, a demonstration).
Visitor: somebody who visits a location or participated in an event.
Presence contact: a person that was present at a location at the same time as a SARS-CoV-2-positive person.
Trace location: a location that was visited by a SARS-CoV-2-positive person.

Overview

The system we present here aims to be easy to use for users, and easy to deploy for venue owners and event organizers. At the same time, it follows the design principles of decentralized contact tracing:

- Privacy by design: no private/sensitive data should be stored in any central database. Moreover, private data should not be derivable from data (e.g., logs) that are stored for operational purposes.

- Purpose limitation by design: abusing the system for population control, location tracing, or other objectives other than sending notifications should be difficult if not impossible.
- Voluntary basis: Use of the digital system is *not mandatory.* A non-digital fallback must always be available.

The system design makes the following assumptions:

1. The digital system requires the use of an app on the user's device[1]
2. Presence tracing should be possible even if the SARS-CoV-2-positive person did not themselves use the app.
3. Users can be trusted to follow notifications from the app, just like they would when notified by a digital contact tracing app.

The first assumption guarantees that the system can notify all visitors, even if the SARS-CoV-2-positive person did not themselves use the app. The second assumption is in line with the current classic approach, which assumes that visitors put their actual phone numbers and name in the attendance logs (there is empirical evidence that many fake numbers are used). In fact, one can reasonably expect better compliance from a privacy-preserving digital solution (where citizens should take action when actually exposed) vs. the classic solutions (where citizens should take action (i.e, write down their real phone number) when attending the venue).

A walkthrough of the system

The following example illustrates how the CrowdNotifier system would work. For illustration, we use a venue as location (e.g., an association's meeting place, a mosque, a church, a meeting room, a bar, a restaurant, a night club), but we recall that a location could also represent an event that spans different locations (e.g., a demonstration).

Setting up. Figure O (on the next page) gives an overview of the proposed digital system. Suppose Charlie manages a venue. To comply with the regulations, she decides to use the system. To do so, she first uses the system to generate and print two QR codes: an entry code and a tracing code. She posts the entry code at the entrance of the room, and keeps the tracing code private. See Figure O, Setup.

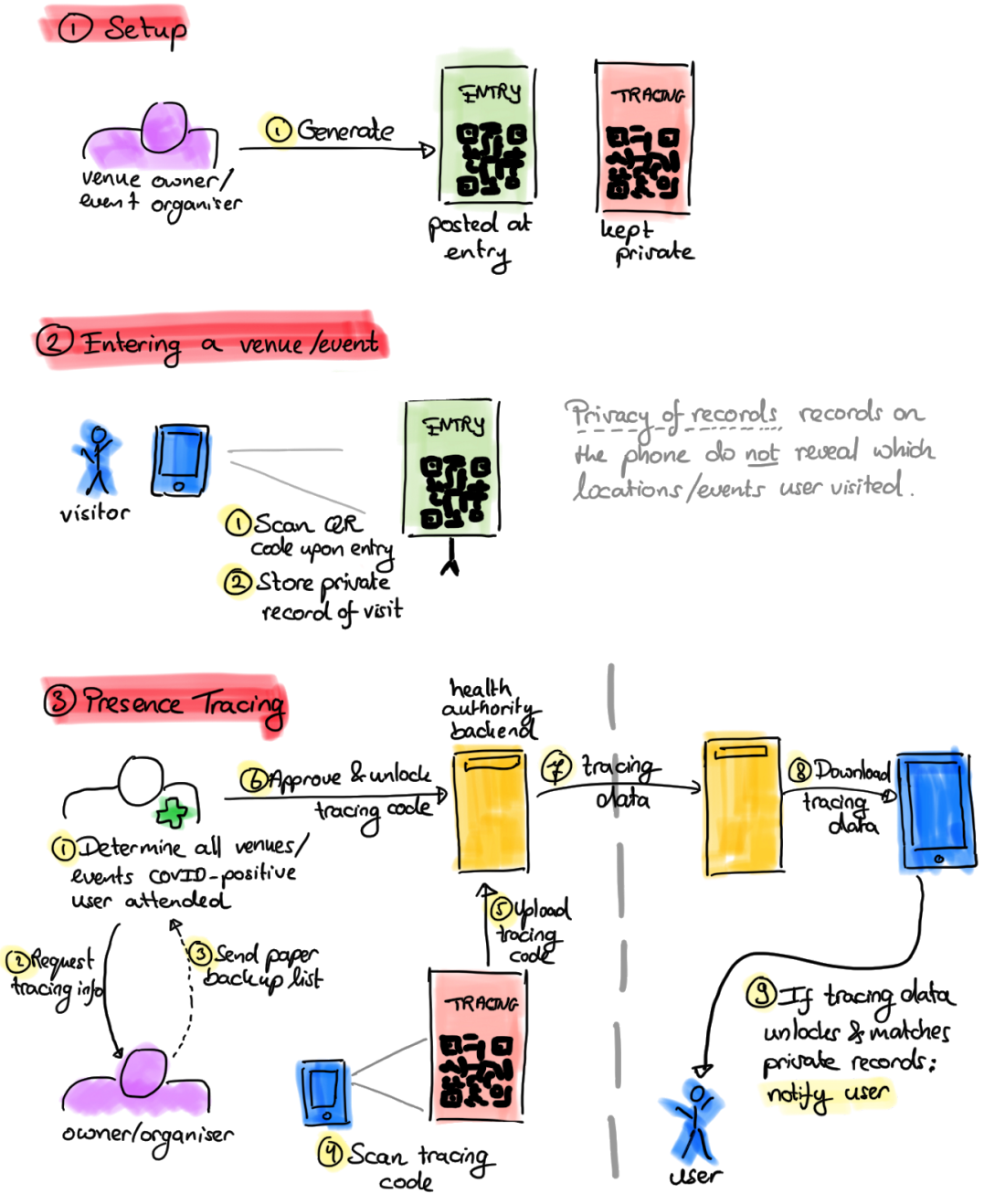
Attending the event/venue. Attendees have a choice to use their app or not. If they do not, they, for instance, write their name and phone number on an attendee list. If they do, they use their app to scan the QR code posted at the entry to the venue. The app shows the name of the venue as a confirmation. See Figure O, Entering a venue/event.

Optionally, the user scans another QR code upon leaving the venue to record the departure time.

[1] This app can be either separate or integrated with existing proximity tracing solutions. In either case, the system must be designed such that the privacy properties of both proximity and presence tracing are guaranteed. (This is the case for the design proposed later on in this document.)

Their phone will store a private record including the time and location visited. (It does not store the name.) Neither the user, nor anyone with access to the user's phone, can use the stored records to determine which location the user visited.

Figure 0: Overview of the core operations in our proposal for Presence Tracing



Presence tracing. Health officials use the traditional interviewing process to determine which events / locations the SARS-CoV-2-positive person attended (step 1). The health official then contacts the owner/organiser for each of these, requesting two things (step 2):

1. The paper backup list (step 3)
2. That the owner/organiser uploads the tracing QR code (steps 4 and 5)

Finally, the health official checks and approves the uploaded tracing data (step 7). This ensures that the data corresponds to the requested venues.

The health authority's backend then disseminates the tracing data (step 7 and 8) to all users of the system. Phones use the provided tracing data to try and unlock private records stored on the phone. If unlocking succeeds, it means that the user was at a location with a SARS-CoV-2-positive person. The phone then determines if there is an epidemiologically relevant overlap between the user's time at the location and that of the SARS-CoV-2-positive person. If there is, the phone notifies the user (step 9).

Requirements and properties

The following is a preliminary list of requirements. This list might be incomplete. Also, the security and privacy requirements make some assumptions on the infrastructure. They are not necessarily complete wrt specific designs that use other infrastructure.

Functional requirements

Following the principles of data-minimization, we require that presence contacts are notified. However, as with proximity tracing, this does not necessarily mean that any party learns the identity or contact information of presence contacts.

F1: Paper fallback option available. The system should support a paper-based fallback option for people that do not have, or do not want to use the smartphone application.

F2: Presence contacts are notified. After a positive SARS-CoV-2 diagnosis for a person, all presence contacts should be notified. Since the SARS-CoV-2-positive user might have used the fallback, notification should be possible even if the SARS-CoV-2-positive user did not use the app (see F1).

Challenge: Capturing exact arrival and departure time might conflict with usability (NF1). While informing the app of arrival should not be complicated, it seems more difficult to require another user action upon departure of a location. Automating departure measurements would seem to require access to sensors that conflict with NF2. Alternatively, the app can ask that users estimate for how long they are planning to stay at a location, or the app simply estimates the departure time (e.g., 2h after arrival or closing time).

F3: Non-presence contacts are not notified. After a positive SARS-CoV-2 diagnosis for a person, any visitors to a location that do not have an epidemiologically relevant overlap with visits of the SARS-CoV-2-positive person are not notified.

Challenge: The need for this requirement depends on judgements made by local health authorities (e.g., taking into account the local situation, the need to be convervative/proactive, and the latest epidemiological insights).

Challenge: this property might be difficult to achieve, depending on how accurate time-keeping is and how accurate SARS-CoV-2-positive visitors (or their devices) can recall their visiting times. The default can be closing time. The app can also poll the user later, asking for the (approximate) departure time of the user. Or the app can ask the user only once it determines a potential overlap with a SARS-CoV-2-positive person.

Non-functional requirements

NF1: Ease of use for users. To ensure adoption, the system should be easy to use from a user's perspective. And in particular no more complicated than base-line paper-based solutions.

This requirement implies that when users have to scan QR codes, these should be small enough (in terms of the amount of data that they contain) that they can be reliably printed/displayed and then scanned by users' devices.

NF2: Easy deployment for locations. To ensure adoption of the system in a large number of locations, the system should not require (special) hardware at locations or server infrastructure managed by locations.

NF3: Speed of notification. After the SARS-CoV-2-positive user has completed the required steps — for example, they informed the contact tracing team of which locations they visited — any presence contacts should receive a notification quickly.

Challenge: background tasks of apps do not necessarily run frequently. While "entering" a location might bring the app to the foreground, and thus cause the OS to assign more background-time; this is not necessarily enough to pick up events days later from background tasks. It might be better therefore to integrate with existing proximity tracing apps.

NF4: Small bandwidth and computation requirements. The system should have a small bandwidth and computation requirement on the side of the user.

Security/Privacy requirements

Privacy of users

PU1: No central collection of personal data. The presence-tracing system should not require the central collection of personal data (e.g., name, IP-address, e-mail address, telephone number, locations visited) of people that visit a location. Nor should the system be able to infer location or co-location data of either visitors or notified visitors.

For the purpose of presence tracing, the health authority is allowed to learn which locations a SARS-CoV-2-positive person visited.

Corollary: this requirement also implies that any central infrastructure used to facilitate the system (e.g., a server) should not be able to deduce that a specific IP address visits a specific location.

PU2: No collection of personal data at a location. The system should not require collecting *at the location itself* personal data (e.g., name, e-mail address, telephone number) of people that visited that location. The sole exception to this is the paper-based fall-back approach.

PU3: No location confirmation attacks given phone. The system should hide which locations somebody has visited. PU1 and PU2 imply that this information should not be available centrally or at the location itself. This information should not be available to attackers that have access to the phone. In particular:

- The UI should not reveal which locations the user visited (e.g., to prevent (retroactive) location tracking by intimate partners and law enforcement).
- The data stored on the phone should not reveal or enable confirmation of visited locations to determined attackers (e.g., law enforcement). Such attackers can gain access to internal storage (i.e., information stored internally by the app), and can also be assumed to have visited specific locations.

Inherent limitation of decentralized systems. In decentralized systems, the phone locally determines if the user should be notified. Therefore, such systems cannot prevent location confirmation attacks with trace locations.

PU4: Do not leak notifications through network traffic. The network traffic (if any) should not reveal that a specific user has been notified.

PU5: Do not leak SARS-CoV-2-positive status through network traffic. The network traffic (if any) should not reveal that a specific user has received a positive test result.

Confidentiality of locations

PL1: Hide which locations had a SARS-CoV-2-positive case from non-visitors. Attackers that never visited a location, should not be able to determine whether that location had a SARS-CoV-2-positive case.

PL2: Hide which locations had a SARS-CoV-2-positive case from non-contemporal visitors. An attacker that visited a location, but not during the time when a SARS-CoV-2-positive person was present, should not be able to determine whether that location had a SARS-CoV-2-positive case.

Challenge: this requirement is more tricky than PL1, as it requires that there is information that is only available to visitors around a specific time. This then implies active displays (or at least changing QR codes) to achieve.

Inherent limitation: No functional presence tracing system can hide that a location is positive from contemporal visitors of that location. This would be in direct violation of the core functional requirement that presence contacts are notified (F1).

PL3: No new central database of locations/events. The system does not create (or rely on) a central database capturing information about all locations. This avoids creating a new database with events (e.g., political and religious gatherings) for the purpose of presence tracing.
Corollary: this requirement also implies that organizers and owners should be able to produce the QR codes anonymously (e.g., using the Tor browser).

Security

S1: No fake notifications (targeting of users). The system should make it impossible to target specific individuals, thereby causing them to be quarantined

S2: No population control (targeting of locations). The system should make it impossible to target sensitive locations (gay bar, biker hangout, bars/places-of-worship in specific neighbourhoods, etc.), thereby causing all the visitors to be quarantined.

Inherent limitation. If an infected person deliberately visits sensitive locations, the functional requirements imply that visitors of this location must be notified. So, at best, we can protect against people that did **not** visit a location.

Confused deputies: Because of requirement (F1), the system must accept locations self-reported by SARS-CoV-2-positive users. Therefore, a malicious SARS-CoV-2-positive user might use the system as a confused deputy to mark a location as a trace location, triggering notifications.

The extent of this problem depends on when the health authorities mark a location as a trace location (e.g., does this require several reports) and whether health authorities can *verify* that the SARS-CoV-2-positive person was at the claimed location.

S3: Automatic dismantling of the system. The dismantling of the system should not depend on central authorities to take action, e.g., to delete keys or data. Instead, as soon as users and location owners stop using the system, any remaining infrastructure should be automatically rendered useless.

Design

The proposed system consists of two parts: (1) a fall-back paper solution and (2) a digital solution using a smartphone application that scans QR codes. The existence of the fallback solution ensures compliance with F1. Both parts are decentralized, but the digital solution achieves better privacy properties.

Design choices

We made several design choices for the digital system to satisfy the above requirements.

Optional action upon departure. For the sake of useability (NF1) and robustness, the system will still function if users do not (or forget to) perform an action when leaving a venue. Instead, the system will use an estimate for the time people usually spend at this venue as default.

(Semi-)Static QR codes for venues. To ensure easy deployment for locations (NF2), venues and events can use static QR codes. As a result of this choice, PL2 becomes impossible to achieve. Users that visited a location at any time, can later confirm that this location was marked as a trace location; even if the user was not there on the same day/time.

If protecting against this attack is essential, venues and organizers can instead generate new QR codes for every day or time-slot (for example, displaying them on screens). Requiring the user to have been present to determine whether the location is notified. Which is the best any system can do.

Single-purpose: notify presence contacts. The app does *not* retain a list of visited locations to aid contact tracers in their interview with a SARS-CoV-2-positive user. Instead — as is currently the norm — contact tracers must rely on interviews with the SARS-CoV-2-positive person to determine which locations that person visited, when and for how long.

The system also does not facilitate contacting or identifying the owner of a location (e.g., a venue owner, or event organizer). Instead — as is currently the norm — contact tracers should rely on public records and information provided by the SARS-CoV-2-positive user to get in touch with the location owner.

The single purpose of the system is to notify presence contacts.

No network connections required when entering a location/event. All necessary information is embedded in the QR code. This ensures the system works even when there is no internet connection. Moreover, the system does not generate any observable network traffic when entering locations as a result.

Regular polling by phones rather than push messages. The use of push notification services such as Firebase requires that users enroll with and connect to 3rd party services. This is not necessary for our use case.

Non goals

Verification of attendance. The system does not aim to provide contact tracers with a proof that a user has been at a location. As a result, users might lie about where they have been, potentially violating S2.

Enforcing quarantines. The system does not aim at providing authorities with the ability to enforce quarantine after notification. We assume that notified users will follow instructions from the health authorities with respect to the actions they should take.

Why not just use the GAEN framework?

Doesn't the GAEN framework already offer precisely what is needed to do this type of colocation tracing?

No — to use the GAEN framework, locations must operate infrastructure (violating NF2). In particular, locations must operate one or more Bluetooth beacons that transmit RPIs as in the GAEN framework. (And, modifications are necessary to ensure that any mobile device inside the location will detect the transmitted RPIs as "close", and to mitigate individuals not inside the location (e.g. in a flat upstairs, or a neighbouring establishment, from too easily registering these modified signals.)

A fallback solution: pen & paper

To satisfy the requirement for a paper-based fallback solution for visitors that do not have or want to use the app (F1), we present a fallback pen & paper solution. This solution also serves as a baseline for the proposed digital solution.

This approach works as follows:

1. A location keeps a box/list per day or time-slot.
2. Every visitor enters contact information (e.g., name and phone number) into the box/list.
3. Only when contact tracing reveals that a SARS-CoV-2-positive person visited a location during a specific time: the authorities receive the corresponding list & contact every presence contact on this list.
4. After 14 days, the boxes/lists are destroyed.

Pros/cons:

+ No central list of information about who visited which location (PU1)
+ Easy to use for users (NF1)
+ Easy to use/deploy for locations (NF2)
+ Fast notification and no bandwidth requirements (NF3 & NF4)
+ No privacy reductions through network traffic (there is none) (PU4&PU5)

+ No information on phone that can be exploited (PU3)
+ Population control requires cooperation of owners/organisers and health authorities (S2)
+ Automatic dismantling (S3)
- Personal data is available locally at the locations (PU2)
- "System"/contact-tracers learn locations of presence contacts (PU1)
- "System"/contact-tracers can target individuals (S1)

Digital presence tracing

The digital presence tracing solution works as follows. We assume that users have the corresponding app installed on their phone.

Setup. The health authority generates a public-private encryption key pair pk_H, sk_H; a cyclic group G of prime order p with generator g; and a public hash function H.

QR Code Generation. A location takes as input the name and location (e.g., address) of the event, and generates a set of QR codes as follows:

1. It picks a random value *salt* and derives a private key *sk* by committing to the name and location by computing
   $sk = H(\ name \ || \ location \ || \ salt\ )$.
   It computes the corresponding public key $pk = g^{sk}$ and it picks a random symmetric key *notification_key*.
2. It constructs three QR codes:
   QR_{entry} = QRCode("entry" || *pk* || *notification_key* || *default_time_in_venue*)
   QR_{exit} = QRCode("exit" || *pk*)
   QR_{trace} = QRCode(Enc(pk_H, *name* || *location* || *salt* || *notification_key*))
   and prints them. The codes QR_{entry} and QR_{exit} are pasted at the entrance(s) of the venue. The code QR_{trace} is kept private by the organizer.

The private key *sk* is needed to perform any type of presence tracing. The data needed to recompute it are stored in an encrypted form in QR_{trace}. Cooperation of the health authority is needed to decrypt these values.

The notification key *notification_key* is used to encode messages by the health authority so that they can only be decrypted by apps of users that actually visited the corresponding location. This protection is not perfect. Anybody that knows or learns *notification_key* can subsequently make it public.

The commitment trick used to compute the private key *sk* ensures that the private key is bound to the name and location of the venue, enabling the health authority to verify that they publish the correct key after decrypting QR_{trace}.

Safely generating the QR codes. It is essential that the salt is only known to the venue owner. Therefore, these QR codes cannot be generated server-side. We propose to use

client-side JavaScript to statelessly generate PDFs containing the QR codes. This PDF should contain three pages:

1. A page that includes the entry QR code QR_{entry}, with clear instructions labeling this as the entry code, explanations of fall backs, and the name of the location.
2. A page that includes the entry QR code QR_{exit}, with clear instructions.
3. A page with the tracing QR code $QR_{trace.}$

The QR code should encode a URL that opens the app directly, without generating further network traffic.

Entering/leaving a venue. Upon entering a venue, the user uses their app to scan the corresponding entry QR code. Let *pk* be the public key in the QR code and *default_time_in_venue* the default time we assume people stay at the venue. The app:

1. Picks a random value *r mod p.*
2. Computes

   $$(g^r, pk^r, Enc(pk, arrival\_time \,||\, notification\_key \,||\, default\_time\_in\_venue))$$

   and stores this tuple together with a label for the current day.

Optionally, upon leaving the venue users can scan the exit QR code, the app behaves as before, but then stores *Enc(pk, departure_time || "departure")* as ciphertext instead.

These steps compute a Diffie-Hellman key exchange between the user's device, holding private key *r* and public key g^r and the venue's public key *pk.* The resulting shared key is pk^r. The user's device only stores the public key g^r and the shared key pk^r.

The stored tuples are constructed in such a way that they hide which venue the user visited to everybody that does not know the private key *sk* corresponding to the public key *pk* of the venue. (Assuming the DDH assumption holds in *G.*)

Devices automatically delete any entry older than 10 days.

Initiating contact tracing. After the contact tracing team of the local health authority has determined that a SARS-CoV-2-positive person has visited a location during the contagious period, they proceed as follows. Let $entry_+$ and $exit_+$ the times that the SARS-CoV-2-positive person entered and exited this location.

1. The contact tracing team creates a case number to manage locations related to this patient.
2. They then contact the owner/organiser of the location and request upload of the data in QR_{trace} to the health authority's server using the given case code. To do so, the owner/organizer scans the QR code with their app, enters the case code, and approves the data upload to the health authority's server.
3. The health authority's server decrypts the uploaded data to recover *name, location, salt* and *notification_key.*
4. The contact tracing team checks that all uploaded information from the tracing codes (in particular, the *name* and *location*) corresponds to the locations the

SARS-CoV-2-positive person visited. They remove any extra uploads under this case code.

5. For each approved upload *name || location || salt || notification_key*, the health backend server recomputes *sk* as above. It encrypts a message c_{msg} to the user using *notification_keys* and publishes the recomputed *sk* together with the corresponding *entry* and *exit* times and c_{msg}.
6. Finally, the owner/organizer generates new QR codes as per Setup above and removes the old codes from the venue.

The check performed in step 4 and the subsequent recomputation of *sk* in step 5 ensures that a malicious or coerced venue owner cannot substitute the QR code of another location without this being detectable by the health authority.

Presence tracing and notification. The user's app regularly (say, every few hours) performs the following checks:

1. The app downloads all (sk, *entry+*, *exit+*, c_{msg}) tuples that were published since the last time it checked.
2. For each (sk, *entry+*, *exit+*, c_{msg}) tuple downloaded, the app proceeds as follows.
    a. It finds all matching entry records for the day corresponding to *entry+* and *exit+*. That is, for every stored tuple *(h, k, c)* (corresponding to the values (g^r, pk^r, *ciphertext)* stored earlier); the app selects records where h^{sk} matches *k*.
    b. For all matching records, the app decrypts the ciphertext *c* using *sk* to recover corresponding entry/exit times, default times in the venue, and the *notification_key*.
    c. The app uses these times (using the default time if no departure time is recorded) to determine if the user's presence at the venue overlaps with the time the SARS-CoV-2-positive user was at the venue. If there is overlap, the app decrypts c_{msg} using the recovered key *notification_key* and notifies the user by displaying this message.

Functionality analysis

F1. See above. Using paper-based fallback solution.

F2 & F3. The digital system keeps track of accurate entry times. Therefore, everybody that potentially overlaps with the SARS-CoV-2-positive user at a location can be notified. Satisfying F2. Moreover, if users scan exit codes (or if default duration of a visit is representative) users will not be notified if they did not overlap with SARS-CoV-2-positive user. Largely satisfying F3.

NF1. Users only need to scan a QR code upon entry (and optionally departure) of a location. No other actions are required.

NF2. Locations do not require additional infrastructure.

NF3. Presence contacts are notified within hours of contact tracers obtaining the required information from the location owner/organizer.

NF4. The bandwidth costs are very small. Less than 256 bytes per location that is marked as a trace location (depending on the length of the supplied message).

Security and privacy analysis

PU1 & PU2. There is no central collection of private data related to users. There is also no data collection at the venue (apart from the fallback method). Therefore PU1 & PU2 are satisfied.

PU3. Provided that the venue owner (or event organiser) and the health authority do not collude, location confirmation attacks against data stored on the phone are not possible as long as a location is not marked as a trace location.

As argued above, the *sk* corresponding to a venue is needed to confirm a location. This value is held in encrypted form by the venue owner, and can only be decrypted by the health authority. Therefore, an attacker needs to coerce (or collude with) both the venue owner and the health authority *and* gain access to a user's phone to perform a confirmation attack.

PU4 & PU5. The system does not generate any sensitive network traffic. All phones download information from the backend server. None of them ever send any information.

PL1. To confirm that a location is a trace location, the adversary needs to know the corresponding public key *pk*. This value is on the QR code at the entrance to the venue. Since non-visitors do not know this value, PL1 is normally satisfied. However, malicious visitors can publish the association between *pk* and a location to enable confirmation attacks.

PL2. PL2 is not satisfied, unless the venue owner generates new QR codes for every day or time-slot.

PL3. PL3 is satisfied. The system does not generate nor rely on the creation of any central database related to venue owners or event organisers.

S1. Individual users cannot be targeted. Any published tracing information will always target all visitors to a specific location.

S2. To target a location requires (1) having access to the private key *sk* corresponding to that location and (2) convincing the health authority to publish it.

The private key *sk* key is *only* available by *decrypting* the information stored in QR_{trace}. This requires coercing/corrupting the venue owner and coercing/corrupting/tricking the health authority.

The protocol has been designed to make it difficult to trick the health authority in publishing any *sk* that does not correspond to a trace location. In particular, the check performed in step 4 ensures that any malicious uploads of QR_{trace} for the wrong venue will be detected.

However, the system in its current form does not protect against SARS-CoV-2-positive people maliciously claiming to have been in a venue in order to get that venue to be marked as a trace location.

S3. The system gracefully dismantles itself as soon as users and/or venue owners and event organizers stop using it. As soon as the users stop scanning QR codes, no new data will be stored by their apps. As a result, old data is automatically deleted after 10 days. From that point onward, users cannot be notified anymore. Uninstalling the app has the same effect.

As soon as venue owners stop generating/displaying QR codes, no visitors to their location/events can be notified anymore. In fact, venue owners can retroactively achieve this by destroying the tracing QR code.

The health authority does maintain the decryption key needed to recover the data stored in the tracing QR code. A malicious operator that fails to destroy this key voluntarily can reduce the guarantees surrounding PU3: the guarantee now only holds as long as the venue owner remains honest or destroyed the tracing QR code.

Beyond this decryption key, all other data is made public. No private or sensitive data is stored centrally. Therefore, the system will gracefully dismantle itself.

Other approaches to presence tracing

We compare the approach proposed in this document with two alternative approaches: a centralized system that tracks which locations users visited; and a decentralized system that uses a central database with location information. Both of these approaches have been deployed by national governments.

Centralized Presence Tracing

Centralized presence tracing systems use a central database to keep track of who visited which locations when. Notification of users then proceeds based on centrally available data.

This approach is for example used by the Singaporean SafeEntry System. Whenever a user enters/exits a location, this location scans either the QR code on the user's TraceTogether token or the barcode on the user's national ID card. Both encode a pseudonym for the user (e.g., in the case of the national ID card this is the card's number). Presumably, this information is shared with backend infrastructure to record entry and departure times.

Evaluation

Because of the centralized design, it is easy to target individuals (violating S1), do crowd control (violating S2) and dismantling requires active actions on behalf of the system operator (violates S3).

Privacy-oriented versions of this design do not need to store personal data on the location (PU2 mostly satisfied), nor on the phone (PU3 satisfied). However, to facilitate notification of presence contacts, the system relies on a central database with extensive personal information (PU1 violated).

Location privacy is also limited. While the system does not necessarily reveal which locations were marked as a trace location (PL1 & PL2 satisfied), the system does rely on a central location database to track who is where (violating PL3).

Abuse-prone Decentralized Presence Tracing

An alternative approach is to build a decentralized presence notification system where users record — on their phone — which locations they visit. No central database of visits is created. Instead, health authorities broadcast identifiers of trace locations whenever they identify them. The user's phone then notifies users.

This approach is for example used as part of the New Zealand NZ Tracer App system. Locations generate a QR code that they place at the entrance of their venue. Visitors scan this QR code with the NZ Tracer App when they enter the venue. The app locally stores a record of the visited location and time of visit.

As part of generating a QR code, venue owners (or managers) need to identify themselves, the venue, and provide contact data. These data are used by contact tracers to contact the venue owners.

Evaluation

Because of the decentralized approach it is not possible to target individuals (S1 satisfied). However, crowd control is still possible, as notification only depends on the central health authority (violating S2). Dismantling also requires active action by the system operator to remove the database of location information (violating S3).

No personal data about users is stored at a central database (satisfying PU1) or the location (satisfying PU2). However, the phone stores a list of the locations that the user has visited, and this list is not protected (violating PU3).

Only a random identifier needs to be published to mark a specific location as a trace location. Thus the identity of trace locations is hidden from non-visitors (satisfying PL1). Non-contemporal visitors of locations, however, can still determine whether a SARS-CoV-2-positive person visited a location that they also visited (even if that visit is not contemporal; violating PL2).

Finally, the current NZ Tracer App system does build a central database of locations and venues, violating PL3. However, the existence of this database is not actually necessary in every decentralized system (instead contact tracers could obtain the required information from the location owner/event organiser directly).

Comparison

In Table C we compare our approach with the two alternative approaches we described above with respect to the security and privacy requirements we identify above.

	Centralized System	Naive Decentralized System	CrowdNotifier
Privacy of Users			
No central data collection (PU1)	❌	✔	✔
No data collection at location (PU2)	✔	✔	✔
No location confirmation attacks given phone (PU3)	✔	❌	✔
No notification side channel (PU4)	unknown	✔	✔
No SARS-CoV-2-positive diagnosis side channel (PU5)	unknown	✔	✔
Confidentiality of locations			

Hide trace locations from non-visistors (PL1)	✔	✔	✔
Hide trace locations from non-contemporal visistors (PL2)	✔	❌/✔[2]	❌/✔[2]
No database of locations (PL3)	❌	❌/✔[3]	✔
Security			
No targeting of individuals (S1)	❌	✔	✔
No crowd control (S2)	❌	❌	✔
Automatic dismantling (S4)	❌	❌	✔

[2]To obtain this property, venue owners and organisers need to generate multiple QR codes, e.g., per event or per day. While the current New Zealand implementation does not support this, comparable systems easily could.
[3]The current New Zealand implementation uses this database. However, the system could function without this database.