CrowdNotifier

Decentralized privacy-preserving presence tracing

Version 5 February 2021

Wouter Lueks (EPFL), Seda Gürses (TU Delft), Michael Veale (UCL), Edouard Bugnion (EPFL), Marcel Salathé (EPFL), Kenneth G. Paterson (ETHZ), 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 presence tracing systems and to present our version, CrowdNotifier. We welcome feedback on the requirements and properties of presence tracing systems we identity as well as the design we propose, 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 close 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 close 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.

To account for airborne transmission in spaces with bad ventilation or physical but non-prolonged contacts, epidemiologists have pointed out the need to also consider people that have been sharing a semi-public location with a SARS-CoV-2-positive person for a prolonged time as contacts to be notified. Such locations include restaurants, bars, classrooms, and places of worship; but also events such as concerts and meetings.

Contact tracing systems, therefore, need means to notify these "presence contacts", i.e., people present a such locations. Digital proximity tracing, such as Exposure Notification, does not provide such means, as its operation means that only people at the venue who were in close physical proximity to the SARS-CoV-2-positive person would be notified.

Existing systems to enable presence tracing 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*, can be used to *effectively notify* any presence contacts of SARS-CoV-2-positive persons, while at the same time providing *strong privacy and abuse-prevention properties*.

Contents

Changelog

Date	Version	Changes
February 5, 2021	v2.1	Full rewrite of the white paper and redesign of the protocol. As a result, replacing of QR codes after notification is no longer necessary. Privacy of visitors of notified locations has been improved. Before, the records on the phone could be linked to the location even if the visitor should not be notified. With the new version, these records remain unlinkable. This version contains a more extensive comparison with existing systems, as well as a more extensive security analysis of CrowdNotifier.
October 8, 2020	v1.0	Initial version

We are grateful for the feedback, questions, suggestions and fixes of everyone that commented on earlier versions of this white paper: Paul-Olivier Dehaye, Daniel Kahn Gillmor, Teo Goddet, Andreas Greulich, Matthias Payer, Thusid, Christian Paul, Trond Arve Wasskog, and Peter Wells.

1 Motivation

Epidemiological studies have recently pointed to the potential of airborne transmission beyond the close physical contact distance of 1–2 meters [9]. While the overall contribution of this transmission route remains unknown, multiple studies have shown that clusters – i.e., events with a comparatively high number of transmission events – play a crucial role in the epidemiological dynamics of the COVID-19 pandemic [1, 12].

Such clusters seem to predominantly occur in crowded indoor environments with limited ventilation. Such environments may include restaurants, bars, classrooms, and places of worship as well as events such as concerts, meetings, and private reunions. For quarantine considerations, contact tracers are thus increasingly including not only individuals who were in close proximity to an infected person, but also those who were present at the same location.

These places and events can host a considerable amount of people, and notifying them thus increases the load of contact tracers. When this load is already high due to the amount of positive cases, manual notification associated to presence can become a bottleneck. Digital presence tracing solutions aim at reducing the load of contact tracers in two dimensions: supporting contact tracers in deciding who to notify, and facilitating the notification process.

The deployment of digital presence solutions requires the creation of infrastructure that inevitably generates new data. The potential misuse of this data raises unease in the population and put at risk the wide adoption of these technologies which in turn reduces their utility as support for contact tracing.

For these reasons, it is important to design digital presence tracing in a privacy-preserving manner, anticipating the dangers associated with mass deployment, to facilitate adoption. In this work, we take into account the following four factors:

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). Deployed 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 infrastructure (e.g., Bluetooth emitters, or coopted smartphone applications [5]) 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 user 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 [3, 6, 7]. 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[1]. 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. Therefore, to ensure that the protection provided by the system is independent of policy decisions, we aim to build a solution that does not rely on such a registry of locations.

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 or 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.

[1] See e.g., in the United Kingdom, The Health Protection (Coronavirus, Collection of Contact Details etc and Related Requirements) Regulations 2020 schedules 1–4.

2 Entities and Terminology

In this white paper we use the following terminology for the entities in a presence-tracing system. All presence tracing systems we examine, whether digital or analog, use the following high-level entities or a subset thereof:

Visitor somebody who visits a location. Sometimes called user. There can be many visitors.

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, an exhibition). There can be many locations.

Location Owner the owner or manager of a location. For simplicity, we assume each location has one owner. An owner may manage many locations.

Health Authority the public health-authority that determines which visitors of which locations to notify. Usually, there is only one health authority.

Backend a backend server that can (potentially) interact with visitors, locations, and the health authority. Usually there is only one backend.

Presence tracing concerns crowds that gather at a certain location at a certain time. This could be a crowd on a Friday evening at a bar, or on a Tuesday afternoon meeting in a community center. For readability we use the term "location" to refer to locations at a specific time as well as specific events (that may happen within one or more physical locations).

Health authorities may use different criteria to determine when to notify visitors to a location for a given time slot (e.g., visitors were wearing masks, there were barriers or large separation between groups). We use the following terminology to indicate locations whose visitors should be notified and the respective visitors:

Trace location a location that was visited by one or more SARS-CoV-2-positive persons, and whose visitors should be notified.

Presence contact a person that was present at a location around the same time as one or more SARS-CoV-2-positive persons and should thus be notified.

3 Overview of CrowdNotifier

We present CrowdNotifier. CrowdNotifier aims to help health authorities to implement presence tracing while being easy to use for users and being easy to deploy for venue owners and event organizers. Moreover, the system aims to be respectful of the following design principles, already implemented in 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.

We design the system under the following assumptions:

1. The health authorities has the means to identify which locations have been visited by a SARS-CoV-2-positive person, and at what times those visits happened.
2. The digital system can rely on the use of a mobile app on the user's device. 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. The design proposed later on in this document fulfills this property.
3. Presence tracing should be possible even if the SARS-CoV-2-positive person did not themselves use the app.
4. Users can be trusted to follow notifications from the app, just like they would when they are notified of a close contact by a digital contact tracing app.

The first assumption implies that CrowdNotifier does not need to provide means to identify which locations had a prolonged visit of a SARS-CoV-2-positive person. Just the means to notify other visitors that this even happened and they are at risk and need to take precautions.

The second assumption delimits the technological capabilities on which the design can rely: the kind of sensors that can be used, and the connectivity and the computational capabilities that can be expected. The use of an app guarantees that the system can notify all visitors, even in the case in which the SARS-CoV-2-positive person did not themselves use the app.

The third assumption supports the voluntary nature of the system. It is in line with the current approach used in many places, which assumes that visitors can put their actual phone numbers and name in an attendance log that can be provided to the authorities. 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).

The fourth assumption implies that there is no need to design an enforcement mechanism to ensure that notified users that receive a notification comply with the instructions they receive.

3.1 A walkthrough of the system

The following example illustrates the operation of the CrowdNotifier system. 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).

In CrowdNotifier users store encrypted records for each visit on their phone. These records

can only be decrypted using the decryption key for the corresponding time slot. Without this key, these records reveal noting, guaranteeing users' privacy against strong attackers. To initiate tracing, the health authority – together with the location owner – computes and publishes the proper time-slot specific decryption keys. Apps regularly download these keys and use them to try to unlock the encrypted records. If encryption succeeds, the user may need to be notified.

To build this time-slot specific encryption system, CrowdNotifier uses an identity-based encryption (IBE) scheme. Each location owner acts as the trusted authority. In this setting, users store records of visits by encrypting against a time-based identity for the location they visit. The location owner with the health authority compute the corresponding identity-based decryption keys to enable tracing.

Setting up. Suppose Charlie manages a location for which she decides to use CrowdNotifier presence notification to help her visitors break transmission chains after an outbreak. To do so, Charlie first uses CrowdNotifier to generate and print two QR codes: an *entry code* and a *tracing code*. The entry code contains information describing the venue as well as an identity-based master public key. The tracing code contains the corresponding master private key.

Charlie posts the entry code in a place where visitors can scan it (e.g., at the entrance, or on the tables), and keeps the tracing code private. See Figure 1, Setup.

Visiting the location. When entering the location, visitors can use the app or not. If they do, they scan the QR code posted at the entry to the location. The app shows the name of the location and asks the user to confirm the check in. See Figure 1, Entering a venue. If they do not use the app, they write their contact information on an attendee list.[2]

Once the app learns the departure time (e.g., because the user explicitly checks out after a reminder from the app), the phone stores a private record containing the identity-based encryption (using the location's master public key) of the entry and departure times, using the entry hour as part of the identity. Our choice of identity-based encryption scheme ensures that these ciphertexts do not reveal the location for which they were created to anyone with access to the user's phone as long as the location owner does not compute and reveal the specific decryption key.

Presence tracing. First, health officials need to determine which locations the SARS-CoV-2-positive person attended (step 1, Figure 1, Presence Tracing), e.g., using the traditional interviewing process in which different positive users report having been at a location. The health official then contacts the owner for each of these locations, requesting two things (step 2):

- The paper backup list (step 3)
- That the location owner uploads tracing keys, i.e., identity-based decryption keys, for the affected time interval (steps 4 and 5).

[2]The collection of non-digital data should also be made according to the data minimization principle. It suffices to provide contact information. For the purpose of notifying visitors, these visitors do not need to be identified.

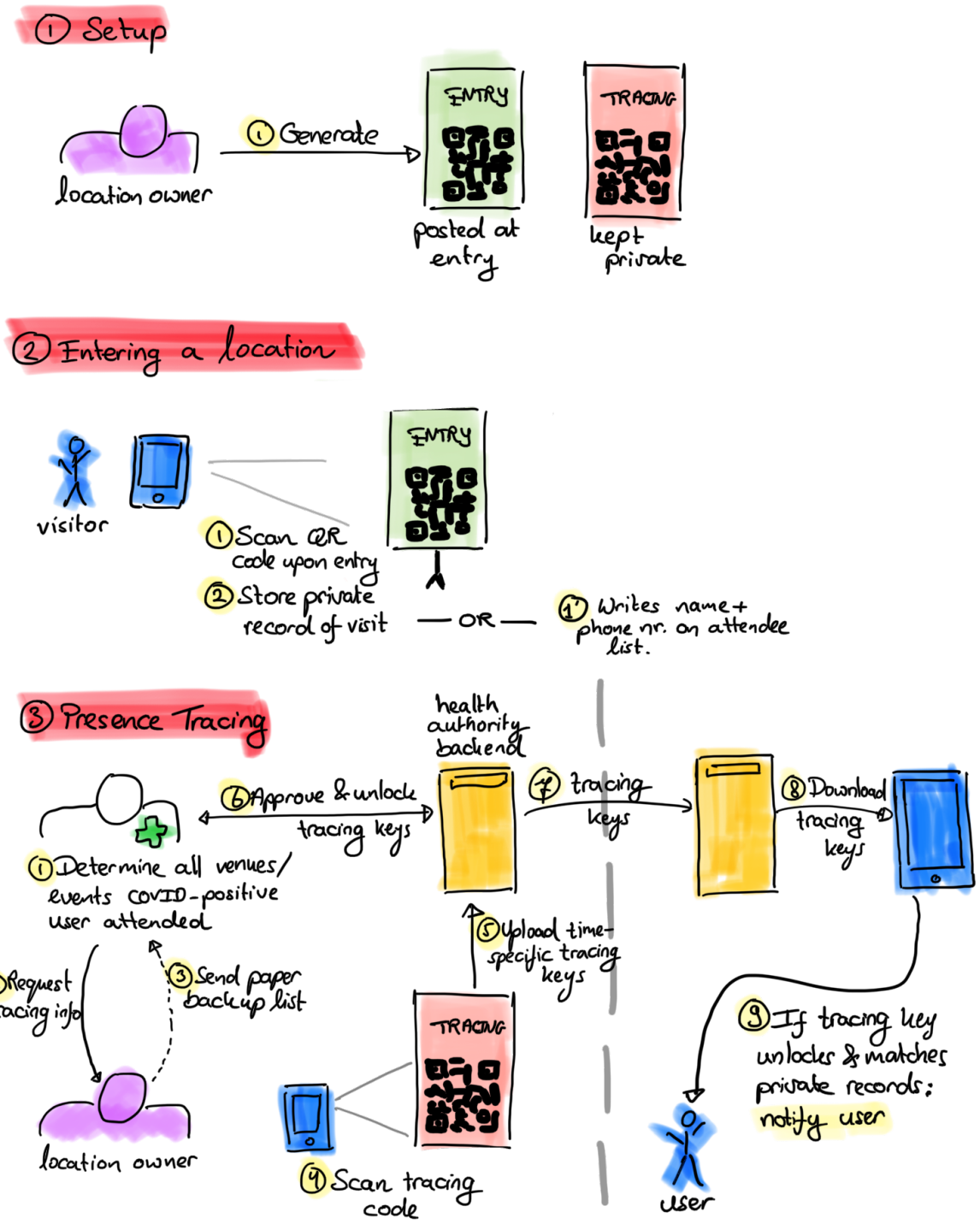

Figure 1: Overview of the core operations in our proposal for Presence Tracing

The health official checks and approves the uploaded tracing keys (step 6). This ensures that the data corresponds to the requested location. Then, the health authority's backend makes available the tracing keys (steps 7 and 8) to all users of the system.

Phones download the provided tracing keys to try and unlock (i.e., decrypt) 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. In this case, the phone recovers the user's entry and departure times for the venue. 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).

4 Presence Tracing Requirements

In this section we describe the requirements that a presence tracing system should fulfil in order to meet epidemiological need while safeguarding privacy, providing checks on power and abuse, and minimising infrastructural impact after an emergency ends.

We stress that we focus on *notification of presence contacts* only. We make the following assumptions:

- The health authority is trusted to (1) determine which locations' visitors should be notified and (2) to trigger notification when required.
- The use of the tool by location owners and visitors is voluntary.
- The system does not need to enforce the adherence of users to their obligations once they are notified.

4.1 Functional requirements

The system must provide the following functionalities.

F1: Analog fallback option available. There should be a non-digital fallback for people that do not have, or do not want to use the smartphone application.

F2: Presence contacts are notified. All presence contacts at a trace location should be notified. Notification should be possible even if the SARS-CoV-2-positive user did not use the app. Following the principles of data-minimization, we do not require that any party learns the identity or contact information of presence contacts.

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 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, set a reminder to ask the user to introduce the leaving time later on, 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 whose visit interval does not have an epidemiologically relevant overlap with visit(s) of the SARS-CoV-2-positive person are not notified. 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).

4.2 Non-functional requirements

To ensure deployability, as well as effectiveness, the system must fulfill the following 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 health authority has determined a trace location, presence contacts at that location should receive a notification quickly.

NF4: Small bandwidth and computation requirements. The system should have a small bandwidth and computation requirement on the side of the user. (And small user bandwidth implies manageable server bandwidth.)

4.3 Security/Privacy requirements

The following defines privacy requirements for visitors and trace locations, and security requirements.

Privacy of users

We consider the following user-privacy requirements.

PU1: No central collection of personal data. The system should not require the central collection of personal data of visitors (e.g., name, IP-address, e-mail address, telephone number, locations visited). Nor should the system be able to infer location or co-location data of either visitors or notified visitors. This requirement implies that the backend server should not be able to deduce that a specific IP address visits a specific location. The health authority, however, needs to learn which locations a SARS-CoV-2-positive person visited.

PU2: No collection of personal data at a location. The digital system should not require the location to collect personal data of visitors (e.g., name, e-mail address, telephone number) at the location. We make an exception for the paper-based fall-back approach.

PU3: No location confirmation attacks. The system should not enable anyone to learn where a user has been. PU1 and PU2 imply that this information should not be available centrally or at the location itself. This information should not be available even if an adversary (1) has access to the user's phone and its internal storage (e.g., law enforcement) and (2) can assume the user to have visited specific locations. In particular:

- The User Interface (UI) should not reveal which locations the user visited (e.g., to prevent location tracking by intimate partners [3, 6, 7] and law enforcement).
- Data stored on the user's phone should not reveal or enable confirmation of visited locations except possibly for when the user should be notified, e.g., when they visited a trace location.

We note that if the phone locally determines whether the user should be notified, it is not possible to prevent location confirmation attacks if a user's visit to a trace locations coincides with the notification period (per F2, the user must be notified in this situation).

PU4: Notification Privacy. No central server or network observer (if any) should be able to determine that a specific user has been notified.

PU5: SARS-CoV-2-positive status privacy. No central party or network observer other than the health authority, should be able to determine if a user has received a positive test result.

Privacy of Locations

We consider the following privacy requirements for locations.

PL1: Hide which locations had a SARS-CoV-2-positive case from non-visitors. Attackers that never visited a location (and do not collude with somebody who did) should not be able to determine whether that location had a SARS-CoV-2-positive case.

Attackers that do visit the location at the right time will learn, as they must be notified (because of the functional requirement F2). Therefore, anyone that colludes with a legitimate visitor will also learn whether a location is a trace location. In the extreme, it is possible to mount a coordinated attack to learn the status of many locations by visiting them.

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 is a trace location.

This requirement implies that there is information that is available to contemporal visitors, that is not available to non-contemporal visitors. CrowdNotifier does not achieve this property as it requires changing QR codes.

PL3: No new central database of locations. The system must not create (or rely on) a central

database of information about locations.

Corollary. This requirement also implies that organizers and owners should be able to produce the QR codes anonymously (e.g., using the Tor browser).

PL4: Hide locations triggering notification through network traffic. No network observe (if any) should be able to determine that a specific location has been asked to upload tracing information.

Security

We consider the following security requirements.

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 difficult to target sensitive locations (gay bar, biker hangout, places of worship, etc.), thereby causing all the visitors to be quarantined.

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 become automatically useless.

Requirement *S2* is not be achievable in two situations. First, if an infected person deliberately visits sensitive locations: the functional requirements imply that visitors of this location must be notified. Second, a malicious diagnosed user might lie to the health authority about their past whereabouts, converting the health authority into a confused deputy that marks a "clean" location as a trace location. The extent of this problem depends on the mechanisms used by the health authorities to mark a location as a trace location (e.g., whether this requires several reports) and whether health authorities can verify that the SARS-CoV-2-positive person was at the claimed location.

4.4 Non goals

CrowdNotifier does not aim to achieve the following 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.

Preventing denial of service attacks. Presence tracing systems are usually susceptible to denial of service attacks by location owners and health authorities. Both of these parties

can prevent visitors from being notified: the health authority can refuse to initiate tracing; and the legitimate owner can refuse to let users scan in (or provide the wrong information) and/or not provide contact information when asked. We leave denial of service attacks out of scope of technological protections. This illustrates the importance of establishing appropriate incentive structures around technological interventions.

5 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.

5.1 Security and Privacy by Design

We first discuss how the design choices behind CrowdNotifier result in some of the requirements in Section 4.3 being achieved, or being impossible to achieve.

On-device Matching. In CrowdNotifier all information is stored on the user devices, which are in charge of matching visited locations to trace locations and notify the user. Thus, by design, user data is neither stored centrally nor at the location (achieving PU1 and PU2).

Local generation of QR codes. Location owners generate their QR codes locally, without connecting to a server. Thus, the system does not require a central database of locations, achieving PL3. We note that in the design above, PL4 is not achieved, but these uploads could be hidden by deploying dummy uploads.

(Semi-)Static QR codes for venues. Locations can use static QR codes, ensuring easy deployment for locations (fulfilling NF2). As a result of this choice – regardless of the underlying cryptography – PL2 becomes impossible to achieve. Users that visited a location at any time know the same information as contemporary visitors, and can thus 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, locations can instead generate new QR codes for every day or time-slot (for example, displaying them on screens). This requires the adversary to have been present at the same time as the SARS-CoV-2-positive user to determine whether the location is notified. We note that this is the best protection any system can provide as, if the adversary shared space with the SARS-CoV-2-positive user, the adversary should be notified (to fulfill F2).

Single-purpose: notify presence contacts. CrowdNotifier does not aim to aid contact tracers in their interview with a SARS-CoV-2-positive user. Thus, CrowdNotifier does not have to retain a readable list of visited locations (fulfilling PU3). 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.

No network connections required when entering a location/event. All necessary information is embedded in the QR code. Thus the system works even when there is no Internet connection (supporting NF1 and NF2). Moreover, the system does not generate any observable network traffic when entering locations as a result (fulfilling PU4, PU5).

No uploads by users. Notified and SARS-CoV-2-positive users do not generate any network traffic or send data to central servers, achieving PU4 and PU5.

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 (violating PU1). This is not necessary to notify contacts (F2).

User-centric dismantling. From a technical perspective, the system also supports dismantling without the collaboration of any authority by design. When users do not scan or owners do not reveal tracing keys, the system ceases its operation, fulfilling S3. However, we note that even though technically this is the case, in practice an authority could legally mandate venues to have the system in place and require scanning from users, effectively keeping the system in place. Even in this case, as the triggering of notifications cannot be monitored and reactions to notifications cannot be enforced, owners and users could still sabotage the system: owners could provide fake tracing keys that do not result in notifications, and users could ignore the notifications.

Not using the Google/Apple Exposure Notification (GAEN) framework. To use the GAEN framework, locations must operate infrastructure (violating NF2). In particular, locations would need to operate one or more Bluetooth beacons that transmit GAEN Bluetooth beacons for the location to be registered as a contact. Moreover, these beacons would need to be crafted to ensure that any mobile device inside the location would consider the location as a close contact, while avoiding false alarms to others that are not in the same place (e.g. others in a flat upstairs, or a neighbouring establishment).

Not using (GPS) location. We also decided against the use of (GPS) location to automatically determine which locations a user visits, or to determine when a user leaves such a location. First, the perceived increase in usability – the user might not need to check in or out anymore – is mitigated by the lack of accuracy. Indoor GPS localization is not reliable enough to distinguish between different floors and rooms; thus requiring user intervention still. Second, the use of location data to automatically check in or out requires continuous access to the user's location and the corresponding fine-grained location permission. Following what happened with contact-tracing apps, we expect users will experience the use of GPS location as extremely privacy invasive.

Security and Privacy by Implementation

The rest of the requirements cannot be achieved just with architectural choices, and rather depend on concrete implementation details. We assume that an attacker can potentially collude with or coerce any of the parties in the system (users, location owners, health authority) and impersonate them when interacting with other parties in the system. If the adversary coerces all entities in the system, they would have the following capabilities:

- Attackers can visit locations and obtain their public QR codes.
- Attackers can modify the QR code of a venue, e.g., by taping a new QR code on top of the old one.
- Attackers can upload any tracing key (e.g., by colluding with an owner that is authorized to upload)
- Attackers might corrupt users' phones to read stored information and modify phone behavior.
- Attackers might monitor network communication between users/venue-owners and backend systems.

If the adversary has all of these capabilities, in their widest sense (e.g., visit all locations or continuously corrupt the phone) some requirements cannot be fulfilled. CrowdNotifier does not aim to protect users in those cases, as there is no technological defense possible. Concretely, we do not aim at protecting from the following three cases.

First, the case of a health authority that decides to trigger notifications from a venue, even though no SARS-CoV-2-positive patient has declared having visited that location. We assume the health authority has process in place to protect from such function creep. We also show that, as long as the health authority only starts triggering procedures for places declared by SARS-CoV-2-positive patients, the health authority cannot trigger notifications without the collaboration of the owner.

Second, the case of an attacker that has *continuous* control over a users' phone. Such an adversary can hide notifications resulting in denial of service, show false notifications, keep records of locations visited by users, and even determine whether they have been notified; and it is not possible to protect against these attacks. We only consider attackers with one-off access to the phone when analyzing the privacy of visited locations. We note that even given access to the phone, the adversary cannot learn information about trace locations that the user did not visit, nor information related to positive users.

Third, if the adversary controls both the health authority and the QR code at the location (i.e., the adversary knows the secret key of the QR code scanned by the user) we cannot protect the user against notifications. The adversary has all the information to trigger a notification at a place where the user has scanned a QR code. By F2, this user should be notified. In other words, the functional requirements imply that no protection against this attack is possible.

In the following, we detail under which adversarial conditions CrowdNotifier can fulfill which of the remaining requirements.

Preventing False Notifications (S1, S2). As QR codes are printed and posted in a location, it is difficult to tailor them to specific visitors. Therefore, it is difficult to single out specific users even if all of the rest of the system protections fail, ensuring S1. When it comes to groups (S2), we aim to prevent attackers from triggering false notifications in the following situations:

- *Trace security.* Users that check-in only to honest venues (i.e., the QR code was generated by an honest owner, and not modified) will not be notified as long as the honest owners do not upload tracing information. This property holds even against attackers that collude with the health authority (assuming that the HA does not compel tracing information from the honest venue).
- *Info binding.* Users that scan arbitrarily modified QR codes will not be notified as long as the health authority is honest and the locations corresponding to the scan are not marked as trace locations.
- *Entry binding.* Modifications of honestly generated QR codes never result in notifications, not even when the health authority initiates tracing for the original locations.

Preserving privacy of Trace Locations (PL1). As a result of the functional requirements, health authorities learn which locations are marked as trace locations. Similarly, users that are notified learn (or could deduce) which locations are marked as trace locations. We aim to provide privacy of trace locations against non-visitors of a location. Non-visitors (that do not collude with visitors) cannot learn whether that location has been notified.

Preserving privacy of visited locations (PU3). Information about visits stored on the phone should not reveal the visited locations to snapshot attackers that gain one-off access to the data stored on a user's phone, i.e., records stored on the phone that do not coincide with the tracing window for a trace location remain unlinkable. This property even holds when an attacker gains access to the data stored by location owners, as long as the health authority does not cooperate.

5.2 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., a phone number or an e-mail address) 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.

We refer to Section 6.2 for an evaluation of this paper-based fallback approach.

5.3 Cryptographic Building Blocks

CrowdNotifier relies on an identity-based encryption scheme to provide its most important properties. In an identity-based encryption scheme, messages can be encrypted against identities without requiring a specific key-pair to be generated for each identity. Instead, a trust authority – in our case a location owner – generates a master public key mpk and a corresponding master private key msk by running the $\mathsf{IBE.KeyGen}$ algorithm. We emphasize that each location has its own corresponding public key mpk.

To encrypt a message m against an identity id under the public key mpk, a party (in our case a visitor) runs $\mathsf{ctxt} \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, m)$. To decrypt this ciphertext, the trust authority (e.g., location) first computes the corresponding identity-based decryption key $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyDer}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$. Given the identity-based decryption key $\mathsf{sk}_{\mathsf{id}}$ we can then decrypt a ciphertext ctxt encrypted under an identity id by running $m \leftarrow \mathsf{IBE.Dec}(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}, \mathsf{ctxt})$.

CrowdNotifier uses use a slight modification of the FullIdent Boneh-Franklin scheme [2]. (The only modification is that the randomness r now also depends on the identity id, which is passed to $\mathsf{IBE.Dec}$ for verification purposes.) This CCA2 secure scheme ensures ciphertexts are robust. They can only be decrypted using the correct identity-based decryption key. Thus protecting against fake notifications (S1, S2). Moreover, this scheme has strong anonymity properties. It ensures that a ciphertext does not reveal any information about the identity under which it was created, nor the corresponding master public key. Thus ensuring privacy of records (PU3).

Finally, CrowdNotifier uses a regular public-key encryption scheme given by the algorithms KeyGen, Enc, and Dec with the usual semantics. CrowdNotifier also uses a (symmetric) authenticated encryption scheme given by the algorithms $\mathsf{AE.Gen}$, $\mathsf{AE.Enc}$, $\mathsf{AE.Dec}$. The key generator algorithm $k \leftarrow \mathsf{AE.Gen}(1^\ell)$ takes as input a security parameter ℓ and outputs a key k. The encryption algorithm $c \leftarrow \mathsf{AE.Enc}(k, m)$ takes as input a key k and a message $m \in \{0,1\}^*$ and outputs a ciphertext c. The decryption algorithm $m \leftarrow \mathsf{AE.Dec}(k, c)$ takes as input a key k and a ciphertext c, and outputs a message m or an error symbol $\perp$.

Throughout ℓ denotes the security parameter in bits. In practice, $\ell = 128$ suffices.

5.4 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 (by running KeyGen) and picks a public hash function H. It also sets up the IBE scheme by running $\mathsf{pp} \leftarrow \mathsf{IBE.CommonSetup}(1^\ell)$. The public parameters pp describe pairing groups G_1, G_2, and G_T of prime order p generated by generators g_1, g_2, g_T such that there exists a bilinear pairing $e : G_1 \times G_2 \to G_T$. The parameters also describe $H_1 : \{0,1\}^* \to G_1$, a hash function mapping strings into group elements of G_1. The health authority publishes pk_H, pp, and H.

QR Code Generation. A location takes as input the name and place (e.g., address) of the

location encoded as a string info and generates a set of QR codes as follows. It computes two IBE key pairs (one for itself, the location, and one for the health authority)

$$(\mathsf{mpk}_L, \mathsf{msk}_L) \leftarrow \mathsf{IBE.KeyGen}(\mathsf{pp}),$$
$$(\mathsf{mpk}_{HA}, \mathsf{msk}_{HA}) \leftarrow \mathsf{IBE.KeyGen}(\mathsf{pp})$$

and computes $\mathsf{mpk} = \mathsf{mpk}_L \cdot \mathsf{mpk}_{HA} \in G_2$. It encrypts the master secret key msk_{HA} for the health authority by creating the ciphertext $\mathsf{ctxt}_{HA} = \mathsf{Enc}(pk_H, \mathsf{msk}_{HA})$. Next, it picks random nonces $\mathsf{nonce}_1, \mathsf{nonce}_2 \in \{0,1\}^{2\ell}$ and a random symmetric key $\mathsf{notifykey} \leftarrow \mathsf{AE.Gen}(1^\ell)$. Let

$$\begin{aligned} \mathsf{ent} &= \mathsf{mpk}, \\ \pi_{\mathsf{ent}} &= (\mathsf{nonce}_1, \mathsf{nonce}_2), \\ \mathsf{mtr} &= (\mathsf{mpk}, \mathsf{msk}_L, \mathsf{info}, \mathsf{nonce}_1, \mathsf{nonce}_2, \mathsf{ctxt}_{HA}). \end{aligned}$$

The values ent and π_{ent} will become part of the entry QR code. The master tracing key mtr is needed to perform any type of presence tracing. Part of the master private key is stored in encrypted form as ctxt_{HA}. Cooperation of the health authority is needed to decrypt this value, and to compute tracing keys.

The location constructs two QR codes

$$\mathsf{QR}_{\mathsf{entry}} = \mathsf{QRCode}(\text{“entry”} \parallel \mathsf{info} \parallel \mathsf{ent} \parallel \pi_{\mathsf{ent}} \parallel \mathsf{notifykey})$$
$$\mathsf{QR}_{\mathsf{trace}} = \mathsf{QRCode}(\mathsf{mtr} \parallel \mathsf{notifykey})$$

and prints them. The code $\mathsf{QR}_{\mathsf{entry}}$ are pasted at the entrance(s) of the location or on individual tables. The code $\mathsf{QR}_{\mathsf{trace}}$ is kept private by the organizer.

The notification key notifykey 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 (or that collude with somebody that did). This protection is not perfect. Anybody that knows or learns notifykey can subsequently make it public.

Safely generating the QR codes. It is essential that the master private key mpk_L is only known to the location 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 two pages:

- A page that includes the entry QR code $\mathsf{QR}_{\mathsf{entry}}$, with clear instructions labeling this as the entry code, explanations of fall backs, and the name of the location.
- A page with the tracing QR code $\mathsf{QR}_{\mathsf{trace}}$.

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 $\mathsf{QR}_{\mathsf{entry}}$. Let info, $\mathsf{ent} = \mathsf{mpk}$, $\pi_{\mathsf{ent}} = (\mathsf{nonce}_1, \mathsf{nonce}_2)$, and notifykey be the values contained therein. Here info contains the name and address of the location. The app shows info to the user and asks for confirmation that the user wants to check in.

Once the user confirms a checkout (e.g. on their own initiative, or after a reminder from the app) the app proceeds as follows. Let the message

$$m = (\texttt{arrival time}, \texttt{departure time}, \mathsf{notifykey})$$

contain the entry and departure time, and the notification key $\mathsf{notifykey}$. For each hour overlapping with the user's presence at the location, the app computes $\mathsf{hourctr}$, the number of hours since UNIX epoch, and proceeds as follows.

1. The app computes the identity corresponding to this hour

$$\mathsf{id} = H(H(\mathsf{info} \parallel \mathsf{nonce}_1) \parallel \mathsf{hourctr} \parallel \mathsf{nonce}_2),$$

2. The app computes the ciphertext $\mathsf{ctxt} \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, m)$ and stores it together with a label for the current day.

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 hour-specific pre tracing keys to the health authority's servers. The location owner scans the tracing QR code $\mathsf{QR}_{\mathsf{trace}}$ containing $\mathsf{mtr} = (\mathsf{mpk}, \mathsf{msk}_{\mathsf{L}}, \mathsf{info}, \mathsf{nonce}_1, \mathsf{nonce}_2, \mathsf{ctxt}_{\mathsf{HA}})$ and the notification key $\mathsf{notifykey}$ and proceeds as follows:
   (a) For each hour $\mathsf{hourctr}$ (in hours since UNIX epoch) overlapping with the time between entry' and exit' the owner computes the corresponding identity

$$\mathsf{id}_{\mathsf{hourctr}} = H(H(\mathsf{info} \parallel \mathsf{nonce}_1) \parallel \mathsf{hourctr} \parallel \mathsf{nonce}_2),$$

   and the partial identity-based decryption key

$$\mathsf{psk}^{\mathsf{L}}_{\mathsf{id},\mathsf{hourctr}} = \mathsf{IBE.KeyDer}(\mathsf{msk}_{\mathsf{L}}, \mathsf{id}_{\mathsf{hourctr}}).$$

   Let the pre-tracing key be $\mathsf{ptr}_{\mathsf{hourctr}} = (\mathsf{id}_{\mathsf{hourctr}}, \mathsf{psk}^{\mathsf{L}}_{\mathsf{id},\mathsf{hourctr}})$.
   (b) The owner uploads the case code, $\pi_{\mathsf{tr}} = (\mathsf{mpk}, \mathsf{nonce}_1, \mathsf{nonce}_2, \mathsf{ctxt}_{\mathsf{HA}})$, all pre-tracing keys $\mathsf{psk}^{\mathsf{L}}_{\mathsf{id},\mathsf{hourctr}}$, and the notification key $\mathsf{notifykey}$ to the health authority's server.
3. The health authority's system processes each upload as follows.
   (a) It decrypts $\mathsf{ctxt}_{\mathsf{HA}}$ to obtain $\mathsf{msk}_{\mathsf{HA}}$.
   (b) For each hour $\mathsf{hourctr}$ it parses $\mathsf{ptr}_{\mathsf{hourctr}}$ as $(\mathsf{id}_{\mathsf{hourctr}}, \mathsf{psk}^{\mathsf{L}}_{\mathsf{id},\mathsf{hourctr}})$. And it computes its part of the identity-based decryption key

$$\mathsf{psk}^{\mathsf{HA}}_{\mathsf{id},\mathsf{hourctr}} = \mathsf{IBE.KeyDer}(\mathsf{msk}_{\mathsf{HA}}, \mathsf{id}_{\mathsf{hourctr}}).$$

and computes the final identity-based decryption key

$$\mathsf{sk}_{\mathsf{id},\mathsf{hourctr}} = \mathsf{psk}^{\mathsf{L}}_{\mathsf{id},\mathsf{hourctr}} \cdot \mathsf{psk}^{\mathsf{HA}}_{\mathsf{id}}.$$

Let $\mathsf{tr}_{\mathsf{hourctr}} = (\mathsf{id}, \mathsf{sk}_{\mathsf{id},\mathsf{hourctr}})$.

(c) It validates the computed tracing key $\mathsf{tr}_{\mathsf{hourctr}} = (\mathsf{id}_h, \mathsf{sk}_{\mathsf{id},\mathsf{hourctr}})$ by first checking that

$$\mathsf{id}_{\mathsf{hourctr}} = H(H(\mathsf{info} \parallel \mathsf{nonce}_1) \parallel \mathsf{hourctr} \parallel \mathsf{nonce}_2).$$

Next, it picks a random message m of sufficient length and computes the ciphertext $\mathsf{ctxt} \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}_{\mathsf{hourctr}}, m)$, and verifies that $\mathsf{IBE.Dec}(\mathsf{sk}_{\mathsf{id},\mathsf{hourctr}}, \mathsf{id}_{\mathsf{hourctr}}, \mathsf{ctxt}) = m$. If any check fails, it removes the upload.

4. The contact tracing team checks that all uploaded information from the tracing codes (in particular, the description of the location contained in info) corresponds to the locations the SARS-CoV-2-positive person visited. They remove any extra uploads under this case code.
5. The contact tracing team specify a message m that should be sent to the notified users.
6. Finally, the health authority server computes $c_{\mathsf{msg}} = \mathsf{AE.Enc}(\mathsf{notifykey}, m)$, the encrypted message for the notified visitors, and for each relevant hour hourctr it publishes a record $(\mathsf{tr}_{\mathsf{hourctr}}, \mathsf{entry}', \mathsf{exit}', c_{\mathsf{msg}})$.

The checks performed in step 3 ensure that the tracing keys $\mathsf{tr}_{\mathsf{hourctr}}$ are correct and correspond to the location specified in info. The check in step 4 ensures that this location is what the health authority requested. Together, these steps ensure that a malicious or coerced venue owner cannot upload tracing information related to 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 $(\mathsf{tr}_{\mathsf{hourctr}}, \mathsf{entry}', \mathsf{exit}', c_{\mathsf{msg}})$ tuples that were published since the last time it checked.
2. For each tuple downloaded $(\mathsf{tr}_{\mathsf{hourctr}}, \mathsf{entry}', \mathsf{exit}', c_{\mathsf{msg}})$ let $\mathsf{tr}_{\mathsf{hourctr}} = (\mathsf{id}_{\mathsf{hourctr}}, \mathsf{sk}_{\mathsf{id},\mathsf{hourctr}})$. The app proceeds as follows.
   (a) For each record ctxt recorded on the days included between entry' and exit', it tries to decrypt it using $\mathsf{IBE.Dec}(\mathsf{id}_{\mathsf{hourctr}}, \mathsf{sk}_{\mathsf{id},\mathsf{hourctr}}, \mathsf{ctxt})$. The app selects records where decryption succeeds (i.e., those not equal to $\perp$).
   (b) For all selected records ctxt, the app uses the plaintext of ctxt to recover the arrival time, departure time and the notification key notifykey. If there is an overlap between the user's time at the location and what is indicated by entry' and exit', the app uses notifykey to decrypt c_{msg} and notifies the user by displaying this message.

Revisiting requirements

With the extra detail about the CrowdNotifier schemes provided in the previous sections, we revisit the open questions regarding the requirements. We refer to Section 5.1 for an overview.

Functional requirements. The digital construction in combination with the paper-based fallback solution ensures F1. The digital system keeps track of accurate entry and exit times. Therefore, everybody that potentially overlaps with the SARS-CoV-2-positive user at a location can be notified. Satisfying F2. Moreover, if users accurately enter departure times, users will not be notified if they did not overlap with SARS-CoV-2-positive user. Largely satisfying F3.

Security. We revisit requirement S2 that it should be difficult to target groups of people. At a high-level, these are satisfied because both the location owner and health authority need to participate to create tracing keys. And because users verify the location they check into, and the health authority verifies the location for which keys are generated, substitution attacks fail. More specifically, in Section 5.1 we identified three technical requirements: trace security, info binding, and entry binding, needed to satisfy S2. We briefly discuss each.

The security of the underlying identity-based encryption (IBE) scheme, ensures that the location's master secret key is needed to compute tracing keys. As long as the location owner is honest, this master secret key is not available to the attacker. Satisfying *trace security*.

The robustness of the IBE scheme ensures that decryption of a ciphertext, i.e., record, fails if the supplied identity id differs from the one used during encryption. Since the app binds id during encryption to the name and address of the location (contained in info), the user will not receive notifications, as long as the health authority does not accept an upload in steps 3 and 4 of "Initiating contact tracing". Any other upload (e.g., for another venue) even with maliciously generated values, will not result in a match. Satisfying *info binding*.

Similarly, any modifications of honestly generated QR codes by other venues do not result in notifications. The user uses all provided values to compute the identities id. So any modification results in different identities. And per robustness of the IBE scheme, ciphertexts encrypted under different identities fail to decrypt. Satisfying *entry binding*.

Location privacy (PL1). We argue why an adversary that did not visit a location (nor colluded with somebody who did) cannot learn which locations are trace locations. The tracing records $(\mathsf{tr}_{\mathsf{hourctr}}, \mathsf{entry}', \mathsf{exit}', c_{\mathsf{msg}})$ are public. But, by assumption the the adversary does not know the master public key mpk of the location nor the notification key $\mathsf{notifykey}$. Since it does not know $\mathsf{notifykey}$, it gains no information from c_{msg}. The time slot also does not help identity the location.

What is left is the value $\mathsf{tr}_{\mathsf{hourctr}} = (\mathsf{id}_{\mathsf{hourctr}}, \mathsf{sk}_{\mathsf{id},\mathsf{hourctr}})$. The specific choice of IBE scheme and the preimage resistance of H ensures that these values do not help distinguish locations to attackers that did not see $\mathsf{QR}_{\mathsf{trace}}$. Thus PL1 is satisfied.

User privacy (PU3). We revisit the user privacy requirement that information stored by the app does not give any information about locations visited by the user. Note the app only stores the IBE ciphertext ctxt and the day. Anonymity of the IBE scheme guarantees that ctxt does not reveal any information about the corresponding master public key (of the location) nor the identity id. Thus ensuring PU3.

6 Comparison with other Presence Tracing Systems

We compare the security and privacy properties of CrowdNotifier with existing to existing and deployed presence tracing systems.

6.1 Classes of Existing Presence-Tracing Systems

We classify existing presence tracing systems based on where they store data related to a user's visit to a venue or event. We identify three ways of storing these data: at the visited location, at a central server, or on the user's device. Depending on how these data are stored, notification happens in different ways.

While we list examples for each of these categories, we are by no means exhaustive. During the early months of the COVID-19 pandemic, a large number of private initiatives aimed at replacing and digitizing pen-and-paper based systems. For example, Chen mentions over 30 different presence tracing systems in New Zealand alone [4] before the government stepped in and provided the NZ COVID Tracer system. All systems provide the following procedures: *setup* of a location, *recording* a visit, *initiating notification*, and *notification* of visitors.

Data stored at the location. These systems store records of visits at the location itself. Paper based sign-in sheets are a good example of such a system.

To set up a location, the location owner creates a blank sign-in sheet. People that visit the location write their contact details and arrival time on this sheet. When the health authorities determine that visitors of this location for a specific time slot should be notified, they initiate notification by requesting the attendance list from the location owner. The authority notifies relevant visitors using the contact information on the sheet.

We also consider in this class ad-hoc location-specific digital attendance lists. For example, location owners recording contact information on a physical device or online record they own, and only they can access.

Data stored at a central server. These systems store records of visits at a central server that manages many locations. The Singaporean SafeEntry System [8] and the Swiss SocialPass [11] are examples of such systems.

To set up a location, the location owner registers with the central service, usually providing a description and contact information of the location. Upon visiting a location, visitors record their visit and contact information at the central server. For instance:

1. Users enters their data on a web site, e.g, by opening the website by scanning a QR code provided by the location, as in the La Rioja COVID system [10]; or
2. Users use a system-specific app to register visits and transmit their contact information, for instance, by scanning an app-specific QR code as in SocialPass; or
3. The location owner records the user's visit into the central system directly, for instance using a special app to scan ID cards of visitors as in SafeEntry.

	Existing Classes of Systems			
	Store at Location	Store at Server	Store at Phone	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	unknown	✓	✓
No SARS-CoV-2-positive diagnosis side channel (PU5)	✓	✓	✓	✓
Confidentiality of locations				
Hide trace locations from non-visitors (PL1)	✓	✓	✓	✓
Hide trace locations from non-contemporal visitors (PL2)	✓	✓	✗/✓	✗/✓
No database of locations (PL3)	✓	✗	✗/✓	✓
Security				
No targeting of individuals (S1)	✗	✗	✓	✓
No crowd control (S2)	✓	✗	✗	✓
Automatic dismantling (S3)	✓	✗	✗	✓

Table 1: Comparison of three classes of existing presence tracing systems – classified by where they store data related to visits – with CrowdNotifier. Whenever a class does not currently achieve a property, but could achieve it, we mark the cell with "✗/ ✓"

To initiate tracing, health authorities request a excerpt of the visitor log from the location owner, and use that information to notify the relevant visitors.

Data stored on user's device. These systems store visits records on the user's device. The New Zealand NZ COVID system and the UK NHS COVID App system are good examples of this.

To set up a location, the location owner registers with the central service and they receive a location identifier, usually in the form of a QR code. When users visit the location, they use the corresponding app to scan this code, and store a local record on their phone. To initiate notification, health authorities find the corresponding location in their database and broadcast the location identifier to all apps. Apps verify locally whether they visited the location, and notify the user if yes.

6.2 Evaluation of Existing Systems

We evaluate the three types of systems introduced above against the requirements from the Section 4. See Table 1 for a summary. Since none of these systems process information about SARS-CoV-2-positive users, PU5 is satisfied for all of them.

Data stored at the location. No data related to visits is stored centrally nor on the user's phone (PU1 and PU3 satisfied). However, visit records are stored locally (violating PU2). The means to notify users is implementation specific (PU4 unknown).

The system only reveals trace locations to contemporary visitors (PL1 and PL2 satisfied), and

does not require a database of locations (PL3 satisfied).

For security, the health authority can easily target individual users (S1 not satisfied), but requires cooperation of the location owner to target crowds (S2 satisfied). The system automatically dismantles itself as soon as owners stop keeping lists, or users stop recording their presence on these lists (S3 satisfied).

Data stored at a central server. 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 notify presence contacts, the system relies on a central database with extensive personal information (PU1 violated).

Location privacy is 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).

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

Data stored on user's device. In this approach, the central database does not store any personal data about users (satisfying PU1) or the location (satisfying PU2). 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. Locations can avoid this by using one-time registrations (PL2 not satisfied by default, but could be).

The current NZ Tracer App and UK NHS COVID App systems do build a central database of locations and venues, violating PL3. However, the existence of this database is not actually necessary in every decentralized system (PL3 not satisfied, but could be).

Finally, 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 requires active action by the system operator to remove the database of location information (violating S3).

References

[1] Dillon Adam, Peng Wu, Jessica Wong, Eric Lau, Tim Tsang, Simon Cauchemez, Gabriel Leung, and Benjamin Cowling. Clustering and superspreading potential of SARS-CoV-2 infections in Hong Kong. *Nature Medicine*, (26):1714–1719, 2020.

[2] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO 2001*, pages 213–229, 2001.

[3] Rahul Chatterjee, Periwinkle Doerfler, Hadas Orgad, Sam Havron, Jackeline Palmer, Diana Freed, Karen Levy, Nicola Dell, Damon McCoy, and Thomas Ristenpart. The Spyware Used in Intimate Partner Violence. In *S&P 2018*, pages 441–458, 2018.

[4] Andrew Tzer-Yeu Chen. How fragmentation can undermine the public health response to COVID-19. *CoRR*, abs/2009.06279, 2020.

[5] Paul-Olivier Dehaye and Joel Reardon. Proximity Tracing in an Ecosystem of Surveillance Capitalism. *CoRR*, abs/2009.06077, 2020. To appear in WPES 2020.

[6] Diana Freed, Jackeline Palmer, Diana Elizabeth Minchala, Karen Levy, Thomas Ristenpart, and Nicola Dell. "A Stalker's Paradise": How Intimate Partner Abusers Exploit Technology. In *CHI 2018*, page 667, 2018.

[7] Karen Levy and Bruce Schneier. Privacy threats in intimate relationships. *J. Cybersecur.*, 6(1), 2020.

[8] Government of Singapore. Safeentry. `https://www.safeentry.gov.sg/` accessed November 30, 2020.

[9] Kimberly A Prather, Linsey C Marr, Robert T Schooley, Melissa A McDiarmid, Mary E Wilson, and Donald K Milton. Airborne transmission of sars-cov-2. *Science*, 370(6514):303, 2020.

[10] La Rioja. Cómo funciona COVID-19 QR, el sistema que deberías usar al entrar a un bar. `https://www.larioja.com/la-rioja/coronavirus/conoce-funciona-covid19-20201125124800-nt.html` accessed November 30, 2020.

[11] SocialPass. `https://www.socialpass.ch/` accessed November 30, 2020.

[12] Felix Wong and James J Collins. Evidence that coronavirus superspreading is fat-tailed. *Proceedings of the National Academy of Sciences*, 117(47):29416–29418, 2020.

A Identity-based encryption

In this appendix we specify the specific identity-based encryption scheme used by CrowdNotifier. In particular, CrowdNotifier uses a slight modification of the FullIdent Boneh-Franklin scheme [2] given by the following algorithms. (The only modification is that the randomness r now also depends on the identity id, which is passed to $\mathsf{IBE.Dec}$ for verification purposes.)

- $\mathsf{pp} \leftarrow \mathsf{IBE.CommonSetup}(1^\ell)$. On input of security parameter ℓ, generate a type III set of bilinear groups G_1, G_2, G_T generated by respectively g_1, g_2, g_T all of prime order p and let $e : G_1 \times G_2 \to G_T$ be the corresponding pairing. Generate the following hash-functions (modeled as random oracles): $H_1 : \{0,1\}^* \to G_1^*$ a hash function mapping points to the group G_1^*, $H_T : G_T \to \{0,1\}^{2\ell}$ mapping group elements from the target group, $H_3 : \{0,1\}^{2\ell} \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^{2\ell}$, and $H_4 : \{0,1\}^{2\ell} \to K$ mapping into the key-space of the authenticated encryption scheme. Setup outputs $\mathsf{pp} = ((G_1, G_2, G_T, g_1, g_2, g_T, p, e), H_1, H_T, H_3, H_4)$.

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.KeyGen}(\mathsf{pp})$. Pick random $\mathsf{msk} \leftarrow \mathbb{Z}_p$ and set $\mathsf{mpk} = g_2^{\mathsf{msk}} \in G_2$. Return $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyDer}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$. On input of a master public key mpk, a master private key msk, and an identity $\mathsf{id} \in \{0,1\}^*$; outputs private key $\mathsf{sk}_{\mathsf{id}} = H_1(\mathsf{id})^{\mathsf{msk}} \in G_1$.
- $\mathsf{ctxt} \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, m)$. On input of a master public key mpk, an identity $\mathsf{id} \in \{0,1\}^*$, and a message m, proceed as follows. Check that $\mathsf{mpk} \in G_2^*$. Pick a random key $x \leftarrow \{0,1\}^{2\ell}$ and compute

$$\begin{aligned} c_1 &= g_2^r, \\ c_2 &= x \oplus H_T(e(H_1(\mathsf{id}), \mathsf{mpk})^r), \\ c_3 &= \mathsf{AE.Enc}(H_4(x), m) \end{aligned}$$

  where $r = H_3(x, m, \mathsf{id})$. Return $\mathsf{ctxt} = (c_1, c_2, c_3)$.
- $m \leftarrow \mathsf{IBE.Dec}(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}, \mathsf{ctxt})$. On input of an identity id, a private key $\mathsf{sk}_{\mathsf{id}}$, and a ciphertext ctxt, proceed as follows. Parse ctxt as (c_1, c_2, c_3) and return $\perp$ if parsing fails. Check that $\mathsf{sk}_{\mathsf{id}} \in G_1^*$, and compute $x' = c_2 \oplus H_T(e(\mathsf{sk}_{\mathsf{id}}, c_1))$ and $m' = \mathsf{AE.Dec}(H_4(x'), c_3)$. Return $\perp$ if $m' = \perp$. Finally, compute $r' = H_3(x', m', \mathsf{id})$ and check that $c_1 = g_2^{r'}$. If this check fails, return $\perp$, otherwise, return m'.