Wouter Lueks*, Seda Gürses, Michael Veale, Edouard Bugnion, Marcel Salathé,
Kenneth G. Paterson, and Carmela Troncoso

CrowdNotifier: Decentralized Privacy-Preserving Presence Tracing

Abstract: There is growing evidence that SARS-CoV-2 can be transmitted beyond close proximity contacts, in particular in closed and crowded environments with insufficient ventilation. To help mitigation efforts, contact tracers need a way to notify those who were present in such environments at the same time as infected individuals. Neither traditional human-based contact tracing powered by handwritten or electronic lists, nor Bluetooth-enabled proximity tracing can handle this problem efficiently. In this paper, we propose CrowdNotifier, a protocol that can complement manual contact tracing by efficiently notifying visitors of venues and events with SARS-CoV-2-positive attendees. We prove that CrowdNotifier provides strong privacy and abuse-resistance, and show that it can scale to handle notification at a national scale.

Keywords: Presence tracing, proximity tracing, privacy-preserving protocols

DOI 10.2478/popets-2021-0074
Received 2021-02-28; revised 2021-06-15; accepted 2021-06-16.

1 Introduction

Contact tracing aims to slow down the spread of pathogens (e.g., SARS-CoV-2). Traditionally, contact tracing is a manual effort where contact tracers interview positively tested individuals in order to determine who their relevant contacts were. What constitutes a relevant contact depends on the transmission modes of the pathogen of interest. In the case of SARS-CoV-2, close physical contact has long been identified as a main transmission mode, giving rise to digital proximity-tracing applications which have been deployed in numerous countries worldwide [18].

Epidemiological studies have recently pointed to the potential of airborne transmission beyond the close physical contact distance of 1–2 meters [24]. 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 [2, 30]. Such clusters seem to predominantly occur in crowded indoor environments with limited ventilation; including restaurants, bars, classrooms, and places of worship as well as events such as concerts, meetings, and private reunions. Contact tracers thus also aim to reach out to individuals that were present at the same event to advise them of their increased risks and how to best take precautions, e.g., by getting tested, reducing contact with others, or even to order quarantines.

Contact tracers need a way to notify these *presence contacts*. A first generation of *presence tracing* solutions requires a user to "check-in" to a location by entering their contact data on a paper or electronic form. These places and events can host many people. Thus, the notification process can become an unbearable load for contact tracers. As a result, the contact data collection not only introduces privacy and abuse risks, but does not help solve the notification problem at scale.

There is a need for presence tracing solutions that can reduce the load of contact tracers by facilitating the notification process. A first thought may be to use solutions for digital proximity tracing, e.g., [5–7, 14, 29], in which people use smartphones' Bluetooth beacons to detect proximity of infected users. But, these Bluetooth-based solutions are thought to detect and notify people at the venue who were in close physical proximity to the infected person. They cannot reliably capture all visitors present at a location or event.

We present CrowdNotifier, a simple presence tracing system that relies on the established pattern of scanning a QR code [10, 20–22, 27] to *effectively notify* any presence contacts of SARS-CoV-2-positive per-

***Corresponding Author: Wouter Lueks:** EPFL, E-mail: wouter.lueks@epfl.ch
Seda Gürses: TU Delft, E-mail: f.s.gurses@tudelft.nl
Michael Veale: UCL, E-mail: m.veale@ucl.ac.uk
Edouard Bugnion: EPFL, E-mail: edouard.bugnion@epfl.ch
Marcel Salathé: EPFL, E-mail: marcel.salathe@epfl.ch
Kenneth G. Paterson: ETH Zürich, E-mail: kenny.paterson@inf.ethz.ch
Carmela Troncoso: EPFL, E-mail: carmela.troncoso@epfl.ch

sons while at the same time providing *strong privacy and abuse-prevention properties.* We designed CrowdNotifier to limit the risks to users, thus hopefully fostering adoption and limiting cheating (e.g., users providing fake data on contact forms). CrowdNotifier's design anticipates the dangers associated with mass deployment by taking into account the following three factors:

Checks on power. Digital presence-tracing systems aim to facilitate the notification of individuals who meet epidemiologically-defined criteria. If presence tracing solutions leave notification decisions at the sole discretion of a central authority, such infrastructure can be easily abused. If the authority decides to, or is compelled to, notify users based on non-epidemiological criteria, then segments of the population can be ordered to quarantine with no additional infrastructure or cost, e.g., those attending particular events or frequenting certain bars. Similar attacks on proximity tracing systems [11, 13, 28], instead require costly investments in infrastructure or technology. Presence tracing systems should thus have safeguards against abuse of power.

Safeguards for privacy. Systems which leak individuals' location histories can be abused in many ways. Leaks to central authorities may result in persecution; leaks to other users, such as perpetrators of intimate partner violence, may exacerbate coercive control [8, 12, 17]. Revealing which locations were visited by SARS-CoV-2-positive people can lead to stigmatization of owners/organisers and visitors. Presence tracing systems should be designed to remove or minimise these risks.

Graceful dismantling. Because presence tracing systems are introduced in a situation of emergency, it is important to ensure that the impact of the system is minimal after this situation ends. The system sunset should not depend on authorities dismantling infrastructure (e.g., by stopping servers, or deleting databases).

In summary, we make the following contributions:

✓ We describe the security, privacy, and anti-abuse properties that a presence tracing system should satisfy; and formalize the security-and privacy properties.

✓ We propose CrowdNotifier, a privacy-friendly and decentralized presence-tracing design. We prove it satisfies the security-oriented properties we have identified.

✓ We evaluate the performance of CrowdNotifier on phones and show that it scales to nation level.

✓ An application building on CrowdNotifier has been deployed in the wild (see https://github.com/CrowdNotifier/documents).

2 Systematization

We first systematize deployed presence-tracing systems.

2.1 Deployed Presence-Tracing Systems

We classify deployed 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 [9] before the government stepped in and provided the NZ COVID Tracer system [21]. 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 [22] and the Swiss SocialPass [27] 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 enter their data on a web site, e.g, opening the website by scanning a QR code provided by the location, as in the La Rioja COVID system [25].

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.
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.

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 [21] and the UK NHS COVID App system [20] 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.

2.2 Systematization of Entities and Actors

All presence tracing systems we examine, use the following high-level parties 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).

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 visitors that are notified:

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.

2.3 Limitations and Challenges

All of the above designs are susceptible to denial of service attacks by location owners and health authorities. Both can prevent visitors from being notified: the health authority can refuse to initiate tracing; and the owner can refuse to let users scan in (or provides wrong information) and/or not provide contact information when asked. We leave denial of service attacks out of scope of technological protections. The lack of technical solutions illustrates the importance of establishing appropriate incentive structures around technological interventions.

Some of these designs require the creation of a registry of locations. Rendering some types of social gatherings, meetings or communities legible to machines, however, may cause societal harm. While many venues already volunteer or are legally compelled to be part of large, 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, presence tracing technology in combination with relevant legislation that demands they formally register to an entity outside their community may in itself pose a threat regardless of the individual privacy protections or safeguards on the power to issue notifications (see e.g., in the United Kingdom [15, schedules 1–4]). As the definition of relevant locations is ill-suited to being limited by technological means, we instead aim to build a solution that does not rely on such a registry of locations.

3 Presence Tracing Requirements

In this section we describe the requirements that a digital presence tracing system should fulfill in order to meet epidemiological need while safeguarding privacy, providing checks on power and abuse, and minimising infrastructural impact after an emergency ends. We then analyze the existing systems with respect to these requirements and see where they fall short.

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

- The health authority is trusted to (i) determine which locations' visitors should be notified and (ii) 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.

3.1 Requirements

Functional requirements. The system must provide the following functionalities.
F1: Fallback option available. There should be a fallback option for people that do not have, or do not want to use, a smartphone to interact with the system.
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 used the fallback option.

We do not require that any party learns the identity or contact information of presence contacts.
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.

Non-functional requirements. To ensure deployability, as well as effectiveness, the system must fulfill the following requirements.
NF1: Ease of use for users. The system should be easy to use from a user's perspective, and no more complicated than paper-based solutions.
NF2: Easy deployment for locations. The system should not require (special) hardware at locations or server infrastructure managed by locations. The solution should not require frequent human actions to maintain the safe functioning of the system.
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.)

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

I. Privacy of users. We consider the following user-privacy requirements.
PU1: No central collection of personal data. No central party should be able to determine personal data of visitors (e.g., name, IP address, device identifier, telephone number, e-mail address, locations visited). As a corollary, the central parties should not be able to infer location or co-location data of either visitors or notified visitors. This corollary 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 system should not require the location to process or collect personal data of visitors.
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. If the system relies on a smart phone application, then requirement PU3 additionally states that location information should not be available even if an adversary has access to the phone and its internal storage (e.g., law enforcement); and can can visit the same locations as the user. In particular:

- The User Interface (UI) should not reveal which locations the user visited (e.g., to prevent location tracking by intimate partners [8, 12, 17] and law enforcement).
- Data stored on the user's phone should not reveal or enable confirmation of visited locations except for the locations and times for which 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 party 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 author-

ity, should be able to determine if a user has received a positive test result.

II. 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.

PL2: Hide which locations had a SARS-CoV-2-positive case from non-contemporal visitors. An attacker that did not visit a location during the time when a SARS-CoV-2-positive person was present (and did not collude with somebody who did) should not be able to determine whether that location is a trace location. This property should hold even if the attacker (or the people they collude with) did visit this location at other times. This property strengthens PL1.

PL1 and PL2 protect locations against stigmatization. This is particularly relevant for locations such as mosques or gay bars where minorities gather [16].

PL2 is the strongest possible location privacy property with respect to visitors. Attackers that do visit the location during a notification window will learn that the location is a trace location, as they must be notified (because of the functional requirement F2).

When venues provide static information to visitors, e.g., a QR code, the best possible property is PL1. PL2 cannot be met: Once an attacker obtains the static information, they can simulate check-ins at any time, and thus receive notifications even if they were not present.

In both cases, attackers can collude or crowd-source data to increase their collective knowledge of trace locations. In systems that only provide PL1, gathering this data requires only one visit per location; and is thus easier to do.

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

PL4: Hide locations triggering notification through network traffic. The network traffic (if any) should not reveal that a specific location has been asked to upload tracing information.

III. 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 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.

3.2 Evaluation of Deployed Systems

We evaluate deployed systems in each of the three categories introduced in Section 2.1 against the requirements from the previous section (summary in Table 1). None of these systems process information about SARS-CoV-2-positive users, so all satisfy PU5.

Data stored at the location. We focus our analysis on paper-based systems. In these systems, 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, but likely happens by communicating contact data to health authorities. Visitors are therefore contacted out of band (PU4 satisfied).

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. Of the three systems we explored for this setting [22, 25, 27] only SafeEntry processes user data at the location. None of

	Store at Location e.g., paper lists	**Store at Server** e.g., [22, 25, 27]	**Store at Phone** e.g., [20, 21]	**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)	✓	✓	✓	✓
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. Security and privacy analysis of deployed systems grouped by where they store data related to visits; and compared with CrowdNotifier. The indication "×/✓" means does not currently achieve, but could with modifications.

them store this data at the location (PU2 mostly satisfied), nor on the phone (PU3 satisfied). However, to notify presence contacts, the system relies on a central database that gives health authorities access to personal data (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 delete visit records and the database of locations (violating S3).

Data stored on user's device. We focus our analysis on the New Zealand and UK systems [20, 21]. In these systems, the central database nor the location store any personal data about users (satisfying PU1 and PU2). The phone stores a list of the locations that the user has visited, and this list is not protected. Apps can increase protection only if venues use one-time registrations and apps do not store additional data (PU3 not satisfied by default, but could be).

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 COVID-19 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, but could be fixed by not creating this database).

Other designs. We expect that other designs will appear following these design paradigms. These designs will have similar security and privacy properties as the systems studied. A quick analysis of two very recent designs confirms the general thrust of Table 1. The Cléa system [26] (published March 2, 2021) follows the store at the phone paradigm; but uses hardware at locations and rotating QR codes to ensure PU3 and PL2. The considerably more complex Luca system [10] (published March 10, 2021) instead follows the store at the server paradigm and uses encryption and distributed trust to protect the centrally stored visitor logs. However, by design, these data are accessible to health authorities (violating PU1). Finally, the functionality of Cléa and Luca requires uploads to a server by SARS-CoV-2-positive users. It is unclear if either design protects these uploads from traffic analysis (PU5 unknown).

4 CrowdNotifier Overview

The analysis of existing solutions indicates the need for a more privacy-friendly and abuse-resistant system. In this section we introduce CrowdNotifier. CrowdNotifier stores records of visits on the user's phone and it uses cryptography to ensure that records are private and that notifying users requires cooperation of the location owner to prevent abuse.

To facilitate easy deployment by location owners (NF2), CrowdNotifier does not require special hardware at locations, and assumes instead owners place static QR codes that their visitors scan. CrowdNotifier provides the best possible location privacy in this setting. It achieves PL1, but not PL2. At the same time, CrowdNotifier achieves the strongest possible user privacy (achieving PU1, PU2, and PU3).

To achieve these properties, each location owner acts as the trusted authority in an identity-based encryption (IBE) scheme. When users visit a location, they store an encrypted record under a time-based identity for the location. To initiate tracing, the location owner publishes the decryption keys for the corresponding notification time slots, enabling phones to decrypt private records and notify their users.

Letting users scan QR codes is the most prevalent approach in deployed presence tracing systems (as well as other systems in the hospitality sector). The high adoption rate of these solutions in several countries leads us to believe that this is an approach that users are familiar with and can use (achieving NF1).

4.1 A Walkthrough of the System

Figure 1 gives an overview of the proposed system.

Setting up. Suppose Charlie manages a location for which she decides to use CrowdNotifier presence notification. 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(A).

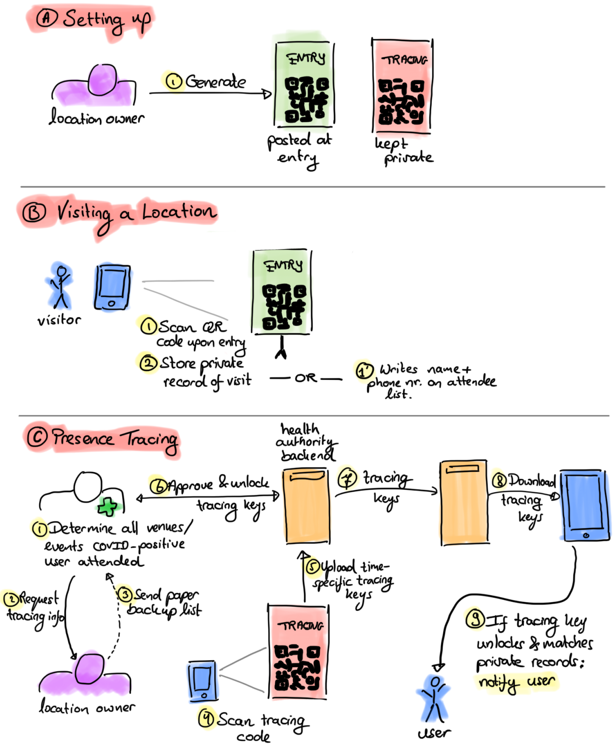
Ⓐ Setting up
location owner
① Generate
ENTRY
posted at entry
TRACING
kept private
Ⓑ Visiting a Location
visitor
ENTRY
① Scan QR code upon entry
② Store private record of visit
— OR —
①' Writes name + phone nr. on attendee list.
Ⓒ Presence Tracing
health authority backend
⑥ Approve & unlock tracing keys
⑦ tracing keys
⑧ Download tracing keys
① Determine all venues/events COVID-positive user attended
② Request tracing info
③ Send paper backup list
⑤ Upload time-specific tracing keys
TRACING
⑨ If tracing key unlocks & matches private records: notify user
location owner
④ Scan tracing code
user

Fig. 1. Overview of our presence-tracing system

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(B). If they do not use the app, they write contact data on an attendee list.

At some point the app learns the check-out time, e.g., after an explicit check out, determined automatically by the app, established by the QR code, or inserted by the user upon a reminder from the app. The app then 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. Anonymity of the IBE scheme ensures that these ciphertexts do not reveal to anyone with access to the phone the location for which they were created as long as the location owner does not reveal the specific decryption key.

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

- The paper backup list (step 3)
- That the owner uploads tracing keys for the affected time interval. The owner scans the tracing

code (step 4), computes the tracing keys (e.g., assisted by a local JavaScript webpage) and uploads them (step 5). These tracing keys take the form of identity-based decryption keys.

The health official checks and approves the uploaded tracing keys (step 6). This ensures that the data corresponds to the requested venue. 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 and 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 a relevant overlap, the phone notifies the user (step 9).

4.2 High-level Security Analysis

We now discuss how CrowdNotifier addresses the security and privacy requirements from Section 3.1 at a high level. In the next two sections, we provide a concrete cryptographic instantiation and a detailed security analysis of the scheme.

4.2.1 Security and Privacy by Design

We first discuss how the design choices behind CrowdNotifier result in some of the requirements in Section 3.1 being achieved, or being impossible to achieve.

On-device matching. In CrowdNotifier all information is stored on the user devices, which match 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 we explain in Section 5.4 how to achieve PL4 through 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 (at the cost of having a person doing these changes for all QR codes displayed or installing screens at the venue). This change would require the adversary to have been present at the same time as the SARS-CoV-2-positive user to determine whether the location is notified (or collude with somebody who was). 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.

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 (facilitating NF1, NF2). Moreover, the system does not generate any observable network traffic when entering locations as a result.

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 to send notifications requires that users enroll with and connect to 3rd party services (violating PU1). This is not necessary to notify contacts (F2).

User- and owner-centric dismantling. By design, the system also supports dismantling without the collaboration of any authority. When users do not scan or owners do not reveal tracing keys, the system ceases its operation, fulfilling S3. 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.

Alternative mechanisms. We considered other mechanisms to detect presence at a location such as Bluetooth Low Energy (BLE) – for example, using the Google/Apple Exposure Notification (GAEN) framework or using custom hardware [3] – or other wireless beacons such as SSIDs. These alternatives, however, require deploying or modifying specific infrastructure and are therefore much harder to deploy at scale (violating NF2). In particular, in the case of BLE, locations would need to operate one or more Bluetooth beacons. Moreover, any wireless beacons would need to be crafted to ensure that all devices inside the location would receive the broadcast, while avoiding false alarms to others that are not in the same place (e.g., others in a flat upstairs, a neighboring establishment, or the next meeting room). This trade-off would make the system less precise than the active check-in process of CrowdNotifier.

We also briefly considered using fine-grained location data to determine a user's location. However, this does not work reliably enough indoor to distinguish different rooms (e.g., consider a block of small meeting rooms), and has negative impact on users' perspective of the system as requesting access to location is perceived as a privacy violation.

4.2.2 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/location-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 a process in place to protect from such function creep. We show that in all cases the health authority requires cooperation of the owner to trigger notifications. 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. 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 with 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. We explain in Section 6 how our construction fulfills the 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). We formalize this property in Definition 4 in Section 6.2.

- *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. We formalized this property in Definition 5 in Section 6.2.
- *Entry binding.* Modifications of honestly generated QR codes never result in notifications, not even when the health authority initiates tracing for the original locations. We formalize this property in Definition 6 in Section 6.2.

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 once visited the location learn (or could deduce) which locations are marked as trace locations and they could tell others. We therefore aim to provide privacy of trace locations against non-visitors of a location that do not collude with visitors. We formalize this property in Definition 7 in Section 6.3.

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. This property must hold even against attackers that can visit locations (e.g., to obtain the QR code) and even when the location the user visited is a trace location at times that do not coincide with the targeted user's visit. We formalize this property in Definition 8 in Section 6.4. In Section 5.4 we show how this property can be made to hold even for attackers that gain access to the private tracing data stored by locations, as long as the attacker does not collude with the health authority.

4.3 Is IBE Needed?

To motivate our choice of using an advanced cryptographic solution, we explain why simpler alternative approaches do not meet the securiy and privacy requirements we established.

First, it is fair to ask: is cryptography necessary at all? Suppose a design where QR codes include a plaintext random venue identifier and phones store this identifier (e.g., as in the UK and New Zealand systems). Then, an attacker that both visits the same location as the user (and thus obtains the random identifier) and obtains the data stored on the user's phone, can confirm the user's visits. This violates PU3, resulting in weak user privacy.

To prevent this attack, user's records on a location must be different from (and uncorrelated to) the record that the adversary would obtain when visiting a venue. The straightforward solution would be to use a standard public key encryption scheme, publish the public key in the QR code, and have the phone store a ciphertext under this key. If encryption fulfills the right anonymity properties, the attacker cannot relate this ciphertext to the key, even if they attacker visited the same venue. This protection, however, falls apart if the visited location is ever marked as a trace location. To facilitate tracing, the location *must* publish the private key. Given the private key, the attacker can confirm a visit, even if the user was not at that location during the notification interval. This again violates PU3.

Another method to uncorrelate the records on the phone from the attacker's view is to use a random identifier, but rotate QR codes frequently, and let phones again store this identifier. For example, Cléa proposes to generate a QR code with a new identifier each day [26]. Thus ensuring PU3 as long as the attacker did not visit the location on the same day as the target. This approach, however, has a high cost. It requires infrastructure at each location to display these QR codes or considerable effort from owners (violating NF2). This approach also relies on perfect operation by the owner/device. If the codes are not rotated, the user's privacy is considerably worse than for CrowdNotifier.

Therefore, CrowdNotifier uses an IBE scheme so that when tracing a location, location owner can produce specific decryption keys *just* for the notification interval. As a result, records created for other intervals cannot be decrypted, and thus remain unlinkable for an attacker. This holds even if notifications are sent for other times, and even if the attacker visited the location or colluded with someone who did; ensuring PU3.

5 The CrowdNotifier System

In this section, we first present the CrowdNotifier cryptographic scheme (Section 5.2) and then show how to use it to build the CrowdNotifier system in Section 5.3.

5.1 Cryptographic Preliminaries

Let G_1, G_2, and G_T be cyclic groups of prime order p generated by generators g_1, g_2, g_T such that exists a bilinear pairing $e : G_1 \times G_2 \to G_T$. Furthermore, let

$H_1 : \{0,1\}^* \to G_1$ be a hash function mapping strings into group elements of G_1.

Let $\mathsf{AE.Gen}, \mathsf{AE.Enc}, \mathsf{AE.Dec}$ be an authenticated encryption scheme. 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 $\perp$.

We use a multi-authority identity-base encryption scheme [23] given by the following algorithms:

- $\mathsf{pp} \leftarrow \mathsf{IBE.CommonSetup}(1^\ell)$. Output common parameters pp.
- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.KeyGen}(\mathsf{pp})$. Generate master public key mpk and master private key msk for a trust authority.
- $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{IBE.KeyDer}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$. On input of master public key mpk, master private key msk, and identity $\mathsf{id} \in \{0,1\}^*$; outputs a decryption key $\mathsf{sk}_{\mathsf{id}}$.
- $\mathsf{ctxt} \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, m)$. On input of a master public key mpk, an identity $\mathsf{id} \in \{0,1\}^*$, and a message m, output a ciphertext ctxt.
- $m \leftarrow \mathsf{IBE.Dec}(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}, \mathsf{ctxt})$. On input of identity id, private key $\mathsf{sk}_{\mathsf{id}}$, and a ciphertext ctxt; either return a message m or a failure symbol $\perp$.

The decryption algorithm takes as input the identity id. We show in Appendix A how the FullIdent scheme by Boneh-Franklin [4] can be modified to include it.

5.2 CrowdNotifier Scheme

A presence-tracing scheme is given by the following methods:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\ell)$. The Setup algorithm, run by the health authority, takes as input a security parameter ℓ, and outputs a set of public parameters pp. All other algorithms implicitly take pp as input.
- $(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{mtr}) \leftarrow \mathsf{GenCode}(\mathsf{info})$. The algorithm $\mathsf{GenCode}$, run by the location owner, takes as input public information info (e.g., the name and address of the location), outputs entry information ent and entry proof π_{ent} (both will be encoded into the entry QR code), as well as master tracing information mtr (which will be encoded into the tracing QR code).
- $\mathsf{rec} \leftarrow \mathsf{Scan}(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{info}, \mathsf{cnt}, \mathsf{aux})$. The algorithm Scan, run by the user, takes as input entry information ent, entry proof π_{ent}, public information info, a counter cnt (representing the time slot during which the user entered), and auxiliary data aux (e.g., entry and departure times); and outputs a record rec.
- $(\mathsf{tr}, \pi_{\mathsf{tr}}) \leftarrow \mathsf{GenTrace}(\mathsf{mtr}, \mathsf{cnt})$. The algorithm $\mathsf{GenTrace}$, run by the location owner, takes master tracing information mtr and a counter cnt (e.g., the time for which to notify visitors) as input; and outputs tracing information tr and a proof of correctness π_{tr}.
- $\mathsf{VerifyTrace}(\mathsf{info}, \mathsf{cnt}, \mathsf{tr}, \pi_{\mathsf{tr}})$. The $\mathsf{VerifyTrace}$ algorithm, run by the health authority, takes as input the venue's public information info, a counter cnt, tracing information tr, and a proof of correctness π_{tr}. It outputs $\top$ if the proof certifies that tr was generated for info and cnt, and $\perp$ otherwise.
- $\mathsf{aux} \leftarrow \mathsf{Match}(\mathsf{rec}, \mathsf{tr})$. The algorithm Match, run by the user, takes as input a record rec and tracing information tr. If tr matches rec, it outputs auxiliary information aux recorded in rec, and $\perp$ otherwise.

A forward secure tracing scheme should satisfy the following correctness property. For all $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\ell)$, $\mathsf{info} \leftarrow \{0,1\}^*$, $(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{mtr}) \leftarrow \mathsf{GenCode}(\mathsf{info})$, $\mathsf{cnt} \in \mathbb{N}$, $(\mathsf{tr}, \pi_{\mathsf{tr}}) \leftarrow \mathsf{GenTrace}(\mathsf{mtr}, \mathsf{cnt})$, $\mathsf{aux} \leftarrow \{0,1\}^*$, and $\mathsf{rec} \leftarrow \mathsf{Scan}(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{info}, \mathsf{cnt}, \mathsf{aux})$ we have that $\mathsf{VerifyTrace}(\mathsf{info}, \mathsf{cnt}, \mathsf{tr}, \pi_{\mathsf{tr}}) = \top$ as well as $\mathsf{Match}(\mathsf{rec}, \mathsf{tr}) = \mathsf{aux}$.

Following the core idea sketched in Section 4.1 we instantiate this scheme by applying an appropriate identity-based encryption scheme.

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\ell)$. On input of security parameter ℓ, run $\mathsf{pp} \leftarrow \mathsf{IBE.CommonSetup}(1^\ell)$, and pick a hash function $H : \{0,1\}^* \to \{0,1\}^{2\ell}$. Output (pp, H).
- $(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{mtr}) \leftarrow \mathsf{GenCode}(\mathsf{info})$. This algorithm is run by the location owner. On input of info, run $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.KeyGen}(\mathsf{pp})$, and pick random nonces $\mathsf{nonce}_1, \mathsf{nonce}_2 \in \{0,1\}^{2\ell}$. Output $\mathsf{ent} = \mathsf{mpk}$, $\pi_{\mathsf{ent}} = (\mathsf{nonce}_1, \mathsf{nonce}_2)$, (embedded in the entry QR code) and $\mathsf{mtr} = (\mathsf{mpk}, \mathsf{msk}, \mathsf{info}, \mathsf{nonce}_1, \mathsf{nonce}_2)$ (embedded in the tracing QR code).
- $\mathsf{rec} \leftarrow \mathsf{Scan}(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{info}, \mathsf{cnt}, \mathsf{aux})$. Run by a visitor. Given entry information $\mathsf{ent} = \mathsf{mpk}$, entry proof $\pi_{\mathsf{ent}} = (\mathsf{nonce}_1, \mathsf{nonce}_2)$, public information info (all from the QR code they scan), counter cnt (e.g., encoding entry hour) and auxiliary data aux (e.g., encoding entry and departure time), compute

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

and output $\mathsf{rec} = \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{aux})$.
- $(\mathsf{tr}, \pi_{\mathsf{tr}}) \leftarrow \mathsf{GenTrace}(\mathsf{mtr}, \mathsf{cnt})$. Run by the location owner to initiate tracing for a time window encoded

in cnt. Parse mtr as $(\mathsf{mpk}, \mathsf{msk}, \mathsf{info}, \mathsf{nonce}_1, \mathsf{nonce}_2)$, and compute $\mathsf{id} = H(H(\mathsf{info} \parallel \mathsf{nonce}_1) \parallel \mathsf{cnt} \parallel \mathsf{nonce}_2)$ and $\mathsf{sk}_{\mathsf{id}} = \mathsf{IBE.KeyDer}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$. Return $\mathsf{tr} = (\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ and $\pi_{\mathsf{tr}} = (\mathsf{mpk}, \mathsf{nonce}_1, \mathsf{nonce}_2)$.
- $\mathsf{VerifyTrace}(\mathsf{info}, \mathsf{cnt}, \mathsf{tr}, \pi_{\mathsf{tr}})$. Run by health authority to verify tr. Parse tr as $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$, and π_{tr} as $(\mathsf{mpk}, \mathsf{nonce}_1, \mathsf{nonce}_2)$. First, verify that $\mathsf{id} = H(H(\mathsf{info} \parallel \mathsf{nonce}_1) \parallel \mathsf{cnt} \parallel \mathsf{nonce}_2)$. Then verify the correctness of $\mathsf{sk}_{\mathsf{id}}$ as follows. Construct $\mathsf{ctxt} = \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, m)$ for a random message $m \in \{0,1\}^{2\ell}$ and check that $\mathsf{IBE.Dec}(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}, \mathsf{ctxt}) = m$. If both check pass, return $\top$ and otherwise $\bot$.
- $\mathsf{aux} \leftarrow \mathsf{Match}(\mathsf{rec}, \mathsf{tr})$. Run by the user. Let $\mathsf{tr} = (\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$. Return $\mathsf{IBE.Dec}(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}, \mathsf{rec})$.

5.3 CrowdNotifier System

The CrowdNotifier system builds on top of the above presence-notification scheme in the following way.

- *Health Authority Setup.* The health authority runs Setup. It publishes the parameters pp, H. The health authority publishes a list $\mathsf{TR}_{\mathsf{list}}$ (initially empty) with tracing information that apps can retrieve.
- *Location Setup and QR code generation.* A venue with public information info (e.g., name of the venue/even, address, etc.) runs $(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{mtr}) \leftarrow \mathsf{GenCode}(\mathsf{info})$ and then constructs two QR codes. The first, $\mathsf{QR}_{\mathsf{entry}}$, encodes the tuple $(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{info})$. The second, $\mathsf{QR}_{\mathsf{trace}}$ encodes mtr. The location places $\mathsf{QR}_{\mathsf{entry}}$ at a place easily accessible by visitors, and keeps $\mathsf{QR}_{\mathsf{trace}}$ private.
- *Visiting a location.* When users visit a location, they use their app to scan $\mathsf{QR}_{\mathsf{entry}}$, let $(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{info})$ be the content. The app shows info to the user, so that they user can check they are scanning in to the correct location. The app aborts if the users does not wants to proceed. Once the app knows the entry and departure times, it encodes them into a string aux. Then for each cnt representing a time slot that overlaps with the user's time at the venue, the app computes and stores $\mathsf{rec} \leftarrow \mathsf{Scan}(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{info}, \mathsf{cnt}, \mathsf{aux})$ and the corresponding day.
- *Initiating notification of users.* Once the health authority determines that it wants to notify the users of a location with public information info during a specific time window, it proceeds as follows. First, it computes the counters $\mathsf{cnt}_1, \ldots, \mathsf{cnt}_t$ corresponding to the time window it wants to notify users. It contacts the location owner, to request and obtain tracing information for each of these counters. Let mtr be the tuple stored in $\mathsf{QR}_{\mathsf{trace}}$. The venue owner runs $(\mathsf{tr}^i, \pi^i_{\mathsf{ptr}}) \leftarrow \mathsf{GenTrace}(\mathsf{mtr}, \mathsf{cnt}_i)$ for $i \in \{1, \ldots, t\}$, and sends the tuples $(\mathsf{tr}^i, \pi^i_{\mathsf{tr}})$ to the health authority. Next the health authority verifies these tuples by running $\mathsf{VerifyTrace}(\mathsf{info}, \mathsf{cnt}_i, \mathsf{tr}^i, \pi^i_{\mathsf{tr}})$ for $i \in \{1, \ldots, t\}$. It aborts, if any check fails. Otherwise, it adds entries $(\mathsf{tr}_i, \text{start time}, \text{end time})$ to $\mathsf{TR}_{\mathsf{list}}$.
- *Notifying visitors.* Apps regularly download entries $(\mathsf{tr}, \text{start time}, \text{end time})$ that have been added to $\mathsf{TR}_{\mathsf{list}}$ since the last time they checked. For each of these entries, and for each record rec stored matching the day of tr, they check whether $\mathsf{Match}(\mathsf{rec}, \mathsf{tr}) \neq \bot$. If so, they extract the user's entry and departure time from aux and compare them against the times published by the health authority. If they overlap, the apps notify their users of a potential exposure.

To reduce the complexity of checking all records, apps automatically expire records that are no longer epidemiologically relevant. For example, in the case of proximity tracing, records usually expire after 10 days.

5.4 Extra Protection

We propose two small extensions to get extra protection.

Protecting mtr. An attacker that has access to the records stored on a user's phone and that gains access to mtr, e.g., by bribing or coercing the location owner, can break record privacy. We propose a modification to the above scheme to require such an attacker to also collude with the health authority to break record privacy.

The idea is straightforward. The honest owner secret shares msk as $\mathsf{msk} = \mathsf{msk}_{\mathsf{L}} + \mathsf{msk}_{\mathsf{HA}}$ and stores $\mathsf{msk}_{\mathsf{L}}$ in the clear, while encrypting $\mathsf{msk}_{\mathsf{HA}}$ (using a CCA2 secure scheme) against the public key of the health authority. To generate tracing information tr, the location owner first sends the ciphertext to the health authority, which decrypts it, to recover $\mathsf{msk}_{\mathsf{HA}}$. Thereafter, they run a distributed version of the $\mathsf{IBE.KeyDer}$ method to compute $\mathsf{sk}_{\mathsf{id}}$ for the identity $\mathsf{id} = H(H(\mathsf{info} \parallel \mathsf{nonce}_1) \parallel \mathsf{cnt} \parallel \mathsf{nonce}_2)$. Such a protocol is easy to construct for the Boneh-Franklin IBE scheme we use.

The CCA2 security of the encryption scheme, and the fact that the health authority itself computes the identity id, ensures that attackers cannot break record privacy, even when giving oracle access to the health authority for other venues.

Hiding uploads by tracing locations. The network traffic patterns generated by uploads by tracing locations are likely recognizable to network adversaries, violating PL4. To protect these uploads, CrowdNotifier can provide an app for location owners. This app then makes regular dummy uploads, for example following an exponential distribution, thereby providing plausible deniability of the real upload.

6 Security & Privacy Properties

The security and privacy of CrowdNotifier relies on properties of the underlying IBE scheme. We define these properties next. All corresponding proofs and IBE games are in Appendix A.

6.1 Properties of IBE Schemes

For the security of our scheme, we require that the underlying IBE scheme satisfies some notion of robustness to ensure that records do not match (i.e., decrypt) under adversarial tracing keys (i.e., decryption keys). In particular, we require the following strengthening of the classical weak robustness notion [1], where the ciphertext is honestly generated, but the adversary can supply master public and private keys.

Definition 1. We say a scheme is *weakly robust under adversarial keys* if for all PPT adversaries $\mathcal{A}$

$$\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{WRAK}}(\ell) = 1\right] < \mathsf{negl}(\ell).$$

See Figure 7 in Appendix A for $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{WRAK}}(\ell)$.

Correctness implies that the adversary can always compute at least one tuple (id, sk) such that $\mathsf{IBE.Dec}$ does not return $\perp$. This robustness property ensures that this is the only (efficiently computable) decryption key that will cause ctxt to successfully decrypt.

We use a slight modification of the FullIdent Boneh-Franklin scheme [4], see Appendix A. The only modification is that the randomness r used to compute ciphertexts now also depends on the identity id, which is passed to $\mathsf{IBE.Dec}$ for verification purposes.

Theorem 1. The (modified) FullIdent scheme by Boneh-Franklin [4] is weakly robust under adversarial keys in the random oracle model for H_T and H_3.

Additionally, we require that ciphertexts have recipient anonymity (they do not reveal the identity under which they have been encrypted) and trusted authority anonymity (they do not reveal under which master public key they have been created). This notion has been formalized as the m-IND-RA-TAA-CPA security property by Paterson and Srinivasan [23]. They also prove that the FullIdent scheme from Boneh and Franklin satisfies this property. Our modification to how r is computed does not violate this property. See Paterson and Srinivasan [23] for the formal security game.

Theorem 2. The (modified) FullIdent Boneh-Franklin IBE scheme is m-IND-RA-TAA-CPA secure.

We also require that an adversary cannot compute master public keys and specific identity id and decryption key $\mathsf{sk}_{\mathsf{id}}$ such that $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ can decrypt two ciphertexts honestly encrypted under different identities id_0 and id_1. In the following definition the adversary controls the messages as well the master public keys $\mathsf{mpk}_0, \mathsf{mpk}_1$. We only require that $\mathsf{IBE.Dec}$ returns a value other than $\perp$.

Definition 2. An IBE scheme has *uniqueness of identity-based decryption keys* if for all PPT adversaries $\mathcal{A}$ we have that

$$\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{UIBDK}}(\ell) = 1\right] < \mathsf{negl}(\ell).$$

See Figure 8 in Appendix A for $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{UIBDK}}$.

Theorem 3. The (modified) FullIdent scheme by Boneh-Franklin [4] has uniqueness of private keys, assuming the collision resistance of H_1, H_T, and H_3.

Finally, we require that an identity-based decryption key $\mathsf{sk}_{\mathsf{id}}$ does not reveal any information about the identity that it is bound to, provided that the adversary does not know the corresponding master public key.

Definition 3. A multi-TA IBE scheme has *decryption-key privacy* if for all PPT adversaries $\mathcal{A}$

$$\Big|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{m-DKPRIV-b}}1(\ell) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{m-DKPRIV-b}}0(\ell) = 1]\Big| < \mathsf{negl}(\ell).$$

See Figure 9 in Appendix A for $\mathsf{Exp}_{\mathcal{A}}^{\text{m-DKPRIV-b}}$.

Theorem 4. The (modified) FullIdent scheme by Boneh-Franklin [4] has decryption-key privacy assuming the DDH assumption holds in G_1 in the random oracle model for H_1.

Next, we show how these IBE properties guarantee the security (Section 6.2) and privacy (Sections 6.3 and 6.4) properties of CrowdNotifier.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{TRSEC}}(\ell)$:
  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\ell})$
  $\mathcal{T} = \emptyset$
  $(\mathsf{info}, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{pp})$
  $(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{mtr}) \leftarrow \mathsf{GenCode}(\mathsf{info})$
  $(\mathsf{tr}', \mathsf{cnt}, \mathsf{aux}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{GetTrace}}(\mathsf{state}, \mathsf{ent}, \pi_{\mathsf{ent}})$
  $\mathsf{rec} \leftarrow \mathsf{Scan}(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{info}, \mathsf{cnt}, \mathsf{aux})$
  If $(\mathsf{cnt}, \mathsf{tr}') \in \mathcal{T}$ or $\mathsf{Match}(\mathsf{rec}, \mathsf{tr}) = \perp$ return 0
  Else return 1

$\mathcal{O}\mathsf{GetTrace}(\mathsf{cnt})$:
  $(\mathsf{tr}, \pi_{\mathsf{tr}}) \leftarrow \mathsf{GenTrace}(\mathsf{mtr}, \mathsf{cnt})$
  $\mathcal{T} = \mathcal{T} \cup \{(\mathsf{cnt}, \mathsf{tr})\}$
  Return tr
```

Fig. 2. Trace Security (TRSEC) experiment.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{INFOBIND}}(\ell)$:
  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\ell})$
  $(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{info}, \mathsf{cnt}, \mathsf{aux}, \mathsf{info}', \mathsf{cnt}', \mathsf{tr}', \pi'_{\mathsf{tr}}) \leftarrow \mathcal{A}(\mathsf{pp})$
  If $(\mathsf{info}, \mathsf{cnt}) = (\mathsf{info}', \mathsf{cnt}')$ return 0
  If $\mathsf{VerifyTrace}(\mathsf{info}', \mathsf{cnt}', \mathsf{tr}', \pi'_{\mathsf{tr}}) = \perp$ return 0
  $\mathsf{rec} \leftarrow \mathsf{Scan}(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{info}, \mathsf{cnt}, \mathsf{aux})$
  If $\mathsf{Match}(\mathsf{rec}, \mathsf{tr}') \neq \perp$ return 1 else 0
```

Fig. 3. Info Binding (INFOBIND) experiment.

6.2 Protection Against False Notifications

The next definition captures that an adversary cannot generate tracing information that matches a honestly-generated QR code. As a result, a malicious health authority cannot generate notifications for honestly-generated QR codes, as long as the owner does not reveal the corresponding tracing information.

Definition 4 (Trace Security)**.** We say that a scheme has *trace security* if for all PPT adversaries $\mathcal{A}$

$$\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{TRSEC}}(\ell) = 1\right] < \mathsf{negl}(\ell),$$

where experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{TRSEC}}(\ell)$ is as defined in Figure 2.

In Appendix B we prove the following theorem.

Theorem 5. The presence tracing scheme has *trace security* provided the underlying IBE scheme is weakly robust under adversarial keys and CPA secure.

The next definition captures that an adversary cannot construct a QR code in such a way that it shows info to the user, while claiming a different value info' towards the health authority when uploading tracing information. As a result, a user that scans in to a venue with info can only be notified if the health authority approved the release of tracing information for info. In the following game, we also let the adversary control the values of cnt (corresponding to the check-in times).

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{ENTRYBIND}}(\ell)$:
  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^{\ell})$
  $(\mathsf{info}, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{pp})$
  $(\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{mtr}) \leftarrow \mathsf{GenCode}(\mathsf{info})$
  $(\pi'_{\mathsf{ent}}, \mathsf{info}', \mathsf{cnt}, \mathsf{aux}, \mathsf{tr}', \pi'_{\mathsf{tr}}, \mathsf{cnt}') \leftarrow \mathcal{A}(\mathsf{state}, \mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{mtr})$
  If $(\pi_{\mathsf{ent}}, \mathsf{info}) = (\pi'_{\mathsf{ent}}, \mathsf{info}')$ return 0
  If $\mathsf{VerifyTrace}(\mathsf{info}, \mathsf{cnt}', \mathsf{tr}', \pi'_{\mathsf{tr}}) = \perp$ return 0
  $\mathsf{rec}' \leftarrow \mathsf{Scan}(\mathsf{ent}, \pi'_{\mathsf{ent}}, \mathsf{info}', \mathsf{cnt}, \mathsf{aux})$
  If $\mathsf{Match}(\mathsf{rec}, \mathsf{tr}') \neq \perp$ return 1 else 0
```

Fig. 4. Entry Binding (ENTRYBIND) experiment.

Definition 5 (Info Binding)**.** We say that a scheme has *info binding* if for all PPT adversarys $\mathcal{A}$ we have that

$$\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{INFOBIND}}(\ell) = 1\right] < \mathsf{negl}(\ell),$$

where experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{INFOBIND}}(\ell)$ is as in Figure 3.

In Appendix B we prove the following theorem.

Theorem 6. The CrowdNotifier scheme has *info binding* provided that the underlying IBE scheme has uniqueness of identity-based decryption keys (see Definition 2) and H is collision resistant.

The following property says that an adversary cannot modify an entry QR code (containing $\mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{info}$) into a related QR code (e.g., by modifying the name in info while keeping ent intact) and still generate notifications provided that the health authority only accepts uploads related to the original info. This property holds even if the adversary has access to mtr.

Definition 6 (Entry binding)**.** We say that a scheme has *entry binding* if for all PPT adversaries $\mathcal{A}$

$$\Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{ENTRYBIND}}(\ell)\right] < \mathsf{negl}(\ell)$$

where experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{ENTRYBIND}}(\ell)$ is as in Figure 4.

Note that in the game in Figure 4 the adversary can modify info' and π'_{ent}, but Scan still takes the original honest value ent as input. Similarly, the health authority verifies tr' against the original value info.

In Appendix B we prove the following theorem.

Theorem 7. The CrowdNotifier scheme has *entry binding* provided that underlying IBE scheme is weakly robust against adversarially chosen keys and hash function H is collision resistant.

```
$\mathsf{Exp}_{\mathcal{A}}^{\text{TRPRIV-b}}(\ell)$:
  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\ell)$
  $\mathcal{C} = \mathcal{T} = \emptyset$
  $(i_0, \mathsf{cnt}_0, i_1, \mathsf{cnt}_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp})$
  $(\mathsf{tr}, \pi_{\mathsf{tr}}) = \mathsf{GenTrace}(\mathsf{mtr}_{i_b}, \mathsf{cnt}_{i_b})$
  $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{state}, \mathsf{tr})$
  If $i_0 \in \mathcal{C}$, $i_1 \in \mathcal{C}$, $(i_0, \mathsf{cnt}_0) \in \mathcal{T}$, or $(i_1, \mathsf{cnt}_1) \in \mathcal{T}$, return 0
  Else return $b'$

$\mathcal{O}\mathsf{AddLocation}(i, \mathsf{info})$:
  $(\mathsf{ent}_i, \pi_{\mathsf{ent}}^i, \mathsf{mtr}_i) \leftarrow \mathsf{GenCode}(\mathsf{info})$
  Return $\top$

$\mathcal{O}\mathsf{GetTrace}(i, \mathsf{cnt})$:
  $\mathcal{T} = \mathcal{T} \cup \{(i, \mathsf{cnt})\}$
  $(\mathsf{tr}, \pi_{\mathsf{tr}}) \leftarrow \mathsf{GenTrace}(\mathsf{mtr}_i, \mathsf{cnt})$
  Return $\mathsf{tr}$

$\mathcal{O}\mathsf{corrupt}(i)$:
  $\mathcal{C} = \mathcal{C} \cup \{i\}$
  Return $(\mathsf{ent}_i, \pi_{\mathsf{ent}}^i, \mathsf{mtr}_i)$
```

Fig. 5. Trace privacy experiment where $\mathcal{O} = \{\mathcal{O}\mathsf{AddLocation}, \mathcal{O}\mathsf{GetTrace}, \mathcal{O}\mathsf{corrupt}\}$.

```
$\mathsf{Exp}_{\mathcal{A}}^{\text{RECPRIV-b}}(\ell)$:
  $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\ell)$
  $\mathcal{C} = \mathcal{T} = \emptyset$
  $(i_0, \mathsf{cnt}_0, \mathsf{aux}_0, i_1, \mathsf{cnt}_1, \mathsf{aux}_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp})$
  If $|\mathsf{aux}_0| \neq |\mathsf{aux}_1|$ return 0
  $\mathsf{rec} \leftarrow \mathsf{Scan}(\mathsf{ent}_{i_b}, \pi_{\mathsf{ent}}^{i_b}, \mathsf{cnt}_i, \mathsf{aux}_i)$
  $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{state}, \mathsf{rec})$
  If $i_0 \in \mathcal{C}$, $i_1 \in \mathcal{C}$, $(i_0, \mathsf{cnt}_0) \in \mathcal{T}$, or $(i_1, \mathsf{cnt}_1) \in \mathcal{T}$, return 0
  Else return $b'$

$\mathcal{O}\mathsf{AddLocation}(i, \mathsf{info})$:
  $(\mathsf{ent}_i, \pi_{\mathsf{ent}}^i, \mathsf{mtr}_i) \leftarrow \mathsf{GenCode}(\mathsf{info})$
  Return $(\mathsf{ent}_i, \pi_{\mathsf{ent}}^i)$
```

Fig. 6. Record privacy experiment where $\mathcal{O} = \{\mathcal{O}\mathsf{AddLocation}, \mathcal{O}\mathsf{GetTrace}, \mathcal{O}\mathsf{corrupt}\}$. $\mathcal{O}\mathsf{GetTrace}$ and $\mathcal{O}\mathsf{corrupt}$ are as in Figure 5.

6.3 Trace Location Privacy

The following definition states that an adversary that never visited a venue cannot distinguish between trace locations based on publicly available tracing information. This property implies that public tracing information does not leak any information to non-visitors.

Definition 7 (Trace location privacy)**.** We say a scheme has *trace location privacy* if for all PPT adversaries $\mathcal{A}$ we have that

$$\big|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{TRPRIV-1}}(\ell) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{TRPRIV-0}}(\ell) = 1]\big| < \mathsf{negl}(\ell)$$

where experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{TRPRIV-b}}$ is defined in Figure 5.

In this game, the adversary can use the $\mathcal{O}\mathsf{AddLocation}$ oracle to create locations with corresponding info strings of its choice. The adversary does not automatically get access to $(\mathsf{ent}_i, \pi_{\mathsf{ent}}^i)$ as these are only available to visitors of locations. Instead, to model a visit, the adversary has to explicitly call $\mathcal{O}\mathsf{corrupt}$ to obtain them (in response, the game also returns mtr_i). The adversary can request tracing information for any location i and counter cnt by calling $\mathcal{O}\mathsf{GetTrace}$.

Finally, as in a standard unlinkability game, the adversary outputs two pairs (i_0, cnt_0) and (i_1, cnt_1). It will receive tracing information for one of them. The game rules out trivial wins where the adversary corrupts the challenge locations i_0 or i_1 (recorded in $\mathcal{C}$) or requested trace information for (i_j, cnt_j) (recorded in $\mathcal{T}$).

In Appendix B we prove the following theorem.

Theorem 8. The scheme has *trace location privacy* in the random oracle model for H assuming the underlying IBE scheme has decryption-key privacy.

6.4 Privacy of Records

The following definition states that no adversary without access to tracing information can link a record rec to a corresponding public QR code. The adversary has full control over where the user checks in, when, and what data is stored in the encrypted record.

Definition 8 (Record Privacy)**.** We say a scheme has *record privacy* if for all PPT algorithms $\mathcal{A}$ we have that

$$\big|\Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{RECPRIV-1}}(\ell) = 1] - \Pr[\mathsf{Exp}_{\mathcal{A}}^{\text{RECPRIV-0}}(\ell) = 1]\big| < \mathsf{negl}(\ell)$$

where experiment $\mathsf{Exp}_{\mathcal{A}}^{\text{RECPRIV-b}}$ is defined in Figure 6.

Theorem 9. The CrowdNotifier scheme has record privacy assuming that the underlying IBE scheme is m-IND-RA-TAA-CPA secure.

Proof sketch. Records are IBE ciphertexts. The challenge ciphertexts/records differ in their master public key (determined by ent_{i_b}), their identity (determined by $\pi_{\mathsf{ent}}^{i_b}$ and cnt_i), and their plaintext payload (determined by aux_b). As the IBE scheme is m-IND-RA-TAA-CPA secure, the adversary cannot exploit these differences: recipient anonymity (RA) hides differences in the identities, trusted authority anonymity (TAA) hides differences in master public keys, and CPA security hides differences in the payload. □

	Laptop i7-7600U	Pixel 2	Huawei P9	Pixel 5	iPhone 6	iPhone 11Pro
	Browser	2017	2016	2020	2014	2019
Scan	11.25 (0.75)	7.02	6.12	5.65	6.38	1.63
GenTrace	1.41 (0.07)	–	–	–	–	–
VerifyTrace	19.1 (0.90)	–	–	–	–	–
Match returning $\perp$	6.60 (0.39)	4.47	3.89	3.59	3.72	0.93
Match returning aux	7.98 (0.40)	5.38	4.63	4.39	4.63	1.15

Table 2. Average run time of CrowdNotifier operations on a laptop CPU implemented in Javascript (standard error of the mean in parentheses), and native implementations on several Android and iOS mobile devices (standard error less than 0.02 ms in all cases).

7 Evaluation

We implemented CrowdNotifier in JavaScript to evaluate the performance for location owners; and Java for Android, and Swift for iOS to evaluate the performance on user's devices. The source code is available via https://github.com/CrowdNotifier/documents. All three implementations use the mcl pairing library [19] which we configured to use the BLS381-12 curve to achieve a security level of roughly 128 bits. We instantiated all random oracles using SHA256 and used the built-in hashAndMapToG1 method to map strings to group elements in G_1.

We evaluated each of the functions Scan, GenTrace, VerifyTrace and Match over 1000 runs. We set the length of aux to 8 bytes to represent two 4-byte timestamps. Table 2 reports the running times. We report two numbers for Match as failing decryptions are almost always caught by the authenticated encryption scheme, thus omitting the need for the extra exponentiation in G_2.

We evaluated the JavaScript implementation on an Intel i7-7600U CPU running at 2.80GHz. Using an x86 build of mcl gives an 8x performance boost. The majority of the time taken for Scan, VerifyTrace, and Match can be attributed to the pairing evaluation (around 6.6 ms per pairing in the JavaScript version of mcl).

Users run Scan and Match on their mobile devices. We used the Android and iOS implementations to evaluate the performance on several mobile devices. We observe that modern iPhones vastly outperform Android devices of similar age. We suspect this is because the Java implementation must traverse the JNI interface on each call to mcl, which adds additional delays.

The cost of running CrowdNotifier on user devices is small, even on old devices. Suppose that users generate 5 records per day; and that the health authorities mark 150 locations as trace locations per day with 3 tracing keys each (corresponding to 3-hour time windows). Then each phone needs to run Match 2,250 times in total. This can be done in within 12 seconds on all devices (and within 3 seconds on recent iPhones).

We want to point out that 150 trace locations per day is a high number (the UK had 226 trace locations from September 2020 to January 2021 [31]). However, if the computation cost becomes prohibitive, apps can easily store a few bits of id to reduce computation time.

Finally, the daily download cost per user is moderate. A tracing key $\mathsf{tr} = (\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ is 32+48 bytes. Therefore, 1000 records require roughly 80 kB. For comparison, a single diagnosed user's keys in proximity tracing systems are roughly the same size as a tracing entry.

8 Take Aways

Notifying contacts of SARS-CoV-2-positive persons is critical to reduce the spread of the pandemic. This includes their close contacts, but also others that shared badly-ventilated events with the positive user. Our contributions show that it is possible to implement this functionality without introducing new infrastructures with high potential for privacy violations and abuse of power.

Acknowledgements

We would like to thank Fabian Aggeler, Martin Alig, Matthias Felix, Johannes Gallmann, and Simon Roesch from Ubique Innovation AG, for the feedback they provided while implementing and deploying the protocols described in this paper. We are also grateful to Dario Fiore and Dennis Jackson for providing feedback on earlier drafts of this paper. Our work also benefited from earlier feedback on the CrowdNotifier White Paper, see the issues of our GitHub repository.

This project has been partially funded by Fondation Botnar.

References

[1] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust Encryption. *J. Cryptol.*, 31(2):307–350, 2018.

[2] 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.

[3] Gilles Barthe, Roberta De Viti, Peter Druschel, Deepak Garg, Manuel Gomez-Rodriguez, Pierfrancesco Ingo, Matthew Lentz, Aastha Mehta, and Bernhard Schölkopf. PanCast: Listening to Bluetooth Beacons for Epidemic Risk Mitigation. *CoRR*, abs/2011.08069, 2020.

[4] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO*, 2001.

[5] Ran Canetti, Yael Tauman Kalai, Anna Lysyanskaya, Ronald L. Rivest, Adi Shamir, Emily Shen, Ari Trachtenberg, Mayank Varia, and Daniel J. Weitzner. Privacy-Preserving Automated Exposure Notification. *IACR ePrint*, 2020:863, 2020.

[6] Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. DESIRE: A Third Way for a European Exposure Notification System Leveraging the best of centralized and decentralized systems. *CoRR*, abs/2008.01621, 2020.

[7] Justin Chan, Dean P. Foster, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham M. Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Sudheesh Singanamalla, Jacob E. Sunshine, and Stefano Tessaro. PACT: Privacy Sensitive Protocols and Mechanisms for Mobile Contact Tracing. *CoRR*, abs/2004.03544, 2020.

[8] 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*. IEEE Computer Society, 2018.

[9] Andrew Tzer-Yeu Chen. How Fragmentation Can Undermine the Public Health Response to COVID-19. *CoRR*, abs/2009.06279, 2020.

[10] culture4life. luca Security Concept. https://luca-app.de/ and https://luca-app.de/securityconcept accessed March 11, 2021.

[11] Paul-Olivier Dehaye and Joel Reardon. Proximity Tracing in an Ecosystem of Surveillance Capitalism. In *WPES*, 2020.

[12] 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.

[13] Rosario Gennaro, Adam Krellenstein, and James Krellenstein. Exposure Notification System May Allow for Large-Scale Voter Suppression. Talk at Real World Crypto 2021.

[14] Google and Apple. Exposure Notifications: Using technology to help public health authorities fight COVID-19, 2020. https://www.google.com/covid19/exposurenotifications/.

[15] The Health Protection (Coronavirus, Collection of Contact Details etc and Related Requirements) Regulations 2020. https://www.legislation.gov.uk/uksi/2020/1005/made.

[16] Nemo Kim. South Korea struggles to contain new outbreak amid anti-gay backlash. https://www.theguardian.com/world/2020/may/11/south-korea-struggles-to-contain-new-outbreak-amid-anti-lgbt-backlash accessed March 11, 2021.

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

[18] Linux Foundation. LF Public Health Landscape. https://landscape.lfph.io/ accessed March 11, 2021.

[19] Shigeo Mitsunari. mcl: a portable and fast pairing-based cryptography library. https://github.com/herumi/mcl.

[20] NHS. NHS COVID-19 app. https://covid19.nhs.uk/, accessed March 11, 2021.

[21] New Zealand Ministry of Health. NZ COVID Tracer app. https://www.health.govt.nz/our-work/diseases-and-conditions/covid-19-novel-coronavirus/covid-19-resources-and-tools/nz-covid-tracer-app, acccessed March 11, 2021.

[22] Government of Singapore. SafeEntry. https://www.safeentry.gov.sg/ accessed March 11, 2020.

[23] Kenneth G. Paterson and Sriramkrishnan Srinivasan. Security and Anonymity of Identity-Based Encryption with Multiple Trusted Authorities. In *Pairing*, 2008.

[24] 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.

[25] 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 March 11, 2020.

[26] Vincent Roca, Antoine Boutet, and Claude Castelluccia. The Cluster Exposure Verification (Cléa) Protocol: Specifications of the Lightweight Version, 2021. https://hal.inria.fr/hal-03146022.

[27] SocialPass. https://www.socialpass.ch/ accessed March 11, 2020.

[28] DP3T Team. Privacy and Security Risk Evaluation of Digital Proximity Tracing Systems, 2020. Version April 21, 2020. Available from https://github.com/DP-3T/documents/blob/master/Security%20analysis/Privacy%20and%20Security%20Attacks%20on%20Digital%20Proximity%20Tracing%20Systems.pdf.

[29] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James R. Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth G. Paterson, Srdjan Capkun, David A. Basin, Jan Beutel, Dennis Jackson, Marc Roeschlin, Patrick Leu, Bart Preneel, Nigel P. Smart, Aysajan Abidin, Seda F. Gürses, Michael Veale, Cas Cremers, Michael Backes, Nils Ole Tippenhauer, Reuben Binns, Ciro Cattuto, Alain Barrat, Dario Fiore, Manuel Barbosa, Rui Oliveira, and José Pereira. Decentralized Privacy-Preserving Proximity Tracing. *CoRR*, abs/2005.12273, 2020.

[30] 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.

[31] Chris Wymant, Luca Ferretti, Daphne Tsallis, Marcos Charalambides, Lucie Abeler-Dörner, David Bonsal, Hinch, Michelle Kendall, Luke Milsom, Matthew Ayres, Chris Holmes, Mark Briers, and Christophe Fraser. The epidemiological impact of the NHS COVID-19 app. *Nature*, 2021.

A Details of IBE Scheme, Games, and Proofs

The modified FullIdent Boneh-Franklin scheme [4] is as follows.

- $\mathsf{pp} \leftarrow \mathsf{IBE.CommonSetup}(1^\ell)$. 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 $H_4 : \{0,1\}^{2\ell} \to \mathcal{K}$ 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 $(\mathsf{mpk}, \mathsf{msk})$.
- $\mathsf{sk_{id}} \leftarrow \mathsf{IBE.KeyDer}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$. On input of a master public key mpk, a master private key msk, and an identity $\mathsf{id} \in \{0,1\}^*$; outputs a private key $\mathsf{sk_{id}} = H_1(\mathsf{id})^{\mathsf{msk}} \in G_1$.
- $\mathsf{ctxt} \leftarrow \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, m)$. 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 $\mathsf{ctxt} = (c_1, c_2, c_3)$.
- $m \leftarrow \mathsf{IBE.Dec}(\mathsf{id}, \mathsf{sk_{id}}, \mathsf{ctxt})$. On input of an identity id, a private key $\mathsf{sk_{id}}$, and a ciphertext ctxt, proceed as follows. Parse ctxt as (c_1, c_2, c_3) and return $\perp$ if parsing fails. Check that $\mathsf{sk_{id}} \in G_1^*$, and compute $x' = c_2 \oplus H_T(e(\mathsf{sk_{id}}, c_1))$ 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'.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{WRAK}}(\ell)$:
  pp ← IBE.CommonSetup($1^\ell$)
  (mpk, id, $m$, id$_0$, $sk_0$, id$_1$, $sk_1$) ← $\mathcal{A}$(pp)
  If $m = \perp$ or (id$_0$, $sk_0$) = (id$_1$, $sk_1$) return 0
  ctxt = IBE.Enc(mpk, id, $m$)
  $m_0$ = IBE.Dec(id$_0$, $sk_0$, ctxt)
  $m_1$ = IBE.Dec(id$_1$, $sk_1$, ctxt)
  If $m_0 \neq \perp \wedge m_1 \neq \perp$ return 1 else 0
```

Fig. 7. Weak Robustness under Adversarial Keys (WRAK) experiment. The adversary's goal is to produce keys and identities such that an honestly generated ciphertext ctxt decrypts under two different keys.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{UIBDK}}(\ell)$:
  pp ← IBE.CommonSetup($1^\ell$)
  (mpk$_0$, id$_0$, $m_0$, mpk$_1$, id$_1$, $m_1$, id, sk$_{id}$) ← $\mathcal{A}$(pp)
  If id$_0$ = id$_1$ return 0
  ctxt$_0$ = IBE.Enc(mpk$_0$, id$_0$, $m_0$)
  ctxt$_1$ = IBE.Enc(mpk$_1$, id$_1$, $m_1$)
  $m_0'$ = IBE.Dec(id, sk$_{id}$, ctxt$_0$)
  $m_1'$ = IBE.Dec(id, sk$_{id}$, ctxt$_1$)
  If $m_0' \neq \perp$ and $m_1' \neq \perp$ return 1 else 0
```

Fig. 8. Uniqueness of Identity-Based Decryption Keys (UIBDK) experiment.

Proof of Theorem 1. We follow the robustness argument of Abdalla et al. [1]. Let $\mathsf{ctxt} = (c_1, c_2, c_3)$. Assume first that $sk_0 \neq sk_1$. Then, since e is non-degenerate and c_1 has full order we have that $e(sk_0, c_1) \neq e(sk_1, c_1)$, and by the collision resistance of H_2, we have that $x_0 = c_2 \oplus H_T(e(sk_0, c_1))$ and $x_1 = c_2 \oplus H_T(e(sk_1, c_1))$ are different as well with overwhelming probability. Let $m_0 = \mathsf{AE.Dec}(H_4(x_0), c_3)$ and $m_1 = \mathsf{AE.Dec}(H_4(x_1), c_3)$ (assuming decryption actually succeeds).

Similarly, if $sk_0 = sk_1$, we must have $\mathsf{id}_0 \neq \mathsf{id}_1$. Therefore, in either case we have that $r_0 = H_3(x_0, m_0, \mathsf{id}_0)$ and $r_1 = H_3(x_1, m_1, \mathsf{id}_1)$ must be different under collision resistance of H_3. So therefore at most one of r_0, r_1 can pass the final check $c_1 = g_2^{r_i}$. □

Proof of Theorem 3. Let $\mathsf{ctxt}_0 = (c_1, c_2, c_3)$ and $\mathsf{ctxt}_1 = (d_1, d_2, d_3)$ and m_0', m_1' be the recovered plaintexts. Let x_0 and x_1 be the original random keys for ctxt_0 respectively ctxt_1. And let $r_0 = H_3(x_0, m_0, \mathsf{id}_0)$ and $r_1 = H_3(x_1, m_1, \mathsf{id}_1)$. Let
$$\begin{aligned} x_0' &= c_2 \oplus H_T(e(\mathsf{sk_{id}}, c_1)) \\ x_1' &= d_2 \oplus H_T(e(\mathsf{sk_{id}}, d_1)) \end{aligned}$$
and $r_i' = H_3(x_i', m_i', \mathsf{id})$ as computed in $\mathsf{IBE.Dec}$. Since $c_1 = g_2^{r_0'} = g_2^{r_0}$ and $d_1 = g_2^{r_1'} = g_2^{r_1}$ we must have $r_i = r_i'$. Therefore, by collision resistance for H_3, we must have

```
$\mathsf{Exp}^{\text{m-DKPRIV-b}}_{\mathcal{A}}(\ell)$:
  $\mathsf{pp} \leftarrow \mathsf{IBE.CommonSetup}(1^{\ell})$
  $\forall i \in [m], (\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{IBE.KeyGen}(\mathsf{pp})$
  $\mathcal{C} = \mathcal{D} = \emptyset$
  $(i_0, \mathsf{id}_0, i_1, \mathsf{id}_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyDer}, \mathcal{O}\mathsf{corrupt}}(\mathsf{pp})$
  $sk_b = \mathsf{IBE.KeyDer}(\mathsf{mpk}_{i_b}, \mathsf{msk}_{i_b}, \mathsf{id}_b)$
  $b' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{KeyDer}, \mathcal{O}\mathsf{corrupt}}(\mathsf{state}, sk_b)$
  If $i_0 \in \mathcal{C}$, $i_1 \in \mathcal{C}$, $(i_0, \mathsf{id}_0) \in \mathcal{D}$, or $(i_1, \mathsf{id}_1) \in \mathcal{D}$, return 0
  Else return $b'$

$\mathcal{O}\mathsf{KeyDer}(i, \mathsf{id})$:
  $\mathcal{D} = \mathcal{D} \cup \{(i, \mathsf{id})\}$
  Return $\mathsf{IBE.KeyDer}(\mathsf{mpk}_i, \mathsf{msk}_i, \mathsf{id})$

$\mathcal{O}\mathsf{corrupt}(i)$:
  $\mathcal{C} = \mathcal{C} \cup \{i\}$
  Return $(\mathsf{mpk}_i, \mathsf{msk}_i)$
```

Fig. 9. The IBE Decryption Key Privacy (m-DKPRIV) experiment

that $(x_i, m_i, \mathsf{id}_i) = (x'_i, m'_i, \mathsf{id})$. Thus, contradicting the assumption that id_0 and id_1 are different. □

Proof sketch of Theorem 4. Decryption keys take the form of $H(\mathsf{id})^{\mathsf{msk}}$ for a master secret key msk. As a result, even seeing many $H(\mathsf{id}), H(\mathsf{id})^{\mathsf{msk}_i}$ pairs, does not allow the adversary to recognize $H(\mathsf{id}_b)^{\mathsf{msk}_b}$. The proof follows by a straightforward encoding of a DDH challenge into sk_b. □

B Proofs of CrowdNotifier Scheme

Proof of Theorem 5. By construction, $\mathsf{rec} = \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{aux})$ where $\mathsf{id} = H(H(\mathsf{info} \parallel \mathsf{nonce}_1) \parallel \mathsf{nonce}_2)$. Suppose $\mathcal{A}$ wins the game. Let $\mathsf{tr} = (\mathsf{id}, sk)$. Then $\mathsf{IBE.Dec}(\mathsf{id}, sk, \mathsf{rec})$ does not output $\perp$. Let $\mathsf{mtr} = (\mathsf{mpk}, \mathsf{msk}, \mathsf{info}, \mathsf{nonce})$. By correctness of the IBE scheme, for decryption key $\mathsf{tr}' = \mathsf{IBE.KeyDer}(\mathsf{mpk}, \mathsf{msk}, \mathsf{id})$, we have that $\mathsf{IBE.Dec}(\mathsf{tr}', \mathsf{rec}) \neq \perp$.

Since the IBE scheme is weakly robust under adversarial keys, we must therefore have that $\mathsf{tr} = \mathsf{tr}'$. The adversary thus recomputed the unique identity-based decryption key tr without querying it from the oracle $\mathcal{O}\mathsf{trace}$. This clearly violates the CPA security of the IBE scheme. □

Proof of Theorem 6. The proof follows directly from the uniqueness of identity-based decryption keys property. Let $\mathsf{tr}' = (\mathsf{id}', \mathsf{sk}'_{\mathsf{id}})$.

First consider Scan. Let $\mathsf{ent} = \mathsf{mpk}$ and $\pi_{\mathsf{ent}} = (\mathsf{nonce}_1, \mathsf{nonce}_2)$; and let $\mathsf{id} = H(H(\mathsf{info} \parallel \mathsf{nonce}_1) \parallel \mathsf{cnt} \parallel \mathsf{nonce}_2)$. Then Scan computes record rec as $\mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{aux})$. Per construction, $\mathsf{Match}(\mathsf{rec}, \mathsf{tr})$ returns the output of $\mathsf{IBE.Dec}(\mathsf{id}', \mathsf{sk}'_{\mathsf{id}}, \mathsf{rec})$ which is by assumption not $\perp$.

Next consider $\mathsf{VerifyTrace}$. On input $\mathsf{tr}' = (\mathsf{id}', \mathsf{sk}'_{\mathsf{id}})$ and $\pi'_{\mathsf{tr}} = (\mathsf{mpk}', \mathsf{nonce}'_1, \mathsf{nonce}'_2)$ it first checks that $\mathsf{id}' = H(H(\mathsf{info}' \parallel \mathsf{nonce}'_1) \parallel \mathsf{cnt}' \parallel \mathsf{nonce}'))$. Because of the collision resistance of H we therefore have with overwhelming probability that $\mathsf{id} \neq \mathsf{id}'$. Next $\mathsf{VerifyTrace}$ computes $\mathsf{ctxt} = \mathsf{IBE.Enc}(\mathsf{mpk}', \mathsf{id}', m)$ for a random message m. Since $\mathsf{VerifyTrace}$ accepts, we must have that $\mathsf{IBE.Dec}(\mathsf{id}', \mathsf{sk}'_{\mathsf{id}}, \mathsf{ctxt}) \neq \perp$.

However, by uniqueness of identity-based decryption keys, $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ cannot simultaneously successfully decrypt the ciphertexts rec and ctxt constructed under different identities id and id'. □

Proof of Theorem 7. Let $\mathsf{ent} = \mathsf{mpk}$ and $\pi'_{\mathsf{ent}} = (\mathsf{nonce}'_1, \mathsf{nonce}'_2)$. Then by construction Scan computes $\mathsf{id} = H(H(\mathsf{info}' \parallel \mathsf{nonce}'_1) \parallel \mathsf{cnt} \parallel \mathsf{nonce}_2)$ and returns record $\mathsf{rec} = \mathsf{IBE.Enc}(\mathsf{mpk}, \mathsf{id}, \mathsf{aux})$. Let $\mathsf{tr} = (\mathsf{id}, \mathsf{sk}_{\mathsf{id}}) = \mathsf{GenTrace}(\mathsf{mtr}, \mathsf{cnt})$, then by correctness $\mathsf{IBE.Dec}(\mathsf{id}, \mathsf{sk}_{\mathsf{id}}, \mathsf{rec}) = \mathsf{aux}$.

Let $\mathsf{tr}' = (\mathsf{id}', \mathsf{sk}'_{\mathsf{id}})$ and $\pi'_{\mathsf{tr}} = (\mathsf{mpk}', \mathsf{nonce}''_1, \mathsf{nonce}''_2)$. Since $\mathsf{VerifyTrace}$ accepts, we must have that $\mathsf{id}' = H(H(\mathsf{info} \parallel \mathsf{nonce}''_1) \parallel \mathsf{cnt}' \parallel \mathsf{nonce}''_2)$. Therefore, since $\mathsf{info} \neq \mathsf{info}'$ by collision resistance of H we must have that $\mathsf{id} \neq \mathsf{id}'$.

Finally, Match does not output $\perp$, and therefore neither does $\mathsf{IBE.Dec}(\mathsf{id}', \mathsf{sk}'_{\mathsf{id}}, \mathsf{rec})$. The fact that rec decrypts under both $(\mathsf{id}, \mathsf{sk}_{\mathsf{id}})$ and $(\mathsf{id}', \mathsf{sk}'_{\mathsf{id}})$ violates the weak robustness under adversarial keys property. □

Proof of Theorem 8. Let $\mathsf{mtr}_{i_b} = (\mathsf{mpk}_b, \mathsf{msk}_b, \mathsf{info}_b, \mathsf{nonce}^b_1, \mathsf{nonce}^b_2)$. Then $\mathsf{tr} = (\mathsf{id}_b, \mathsf{sk}_b)$ where $\mathsf{id}_b = H(H(\mathsf{info}_b \parallel \mathsf{nonce}^b_0) \parallel \mathsf{cnt}_b \parallel \mathsf{nonce}^b_1)$ and $\mathsf{sk}_b = \mathsf{IBE.KeyDer}(\mathsf{mpk}_b, \mathsf{msk}_b, \mathsf{id}_b)$.

To prove the theorem we use game hopping. First, notice that the adversary never corrupts i_0 or i_1 and thus does not learn the corresponding nonces $\mathsf{nonce}^0_1, \mathsf{nonce}^1_1, \mathsf{nonce}^0_2, \mathsf{nonce}^1_2$ directly. Moreover, in the random oracle model for H, the traces for i_0, i_1 for other counters it queried from $\mathcal{O}\mathsf{trace}$ do not reveal any information about the nonces either. Therefore we can replace id_0 with id_1 in the challenge trace tr without the adversary realizing.

Next, we replace sk_0 by sk_1 in the challenge trace. Any adversary that distinguishes this situation breaks the decryption-key privacy experiment. This completes the proof. □