CrowdNotifier

Deployment and Implementation Notes

Version April 7, 2021

The CrowdNotifier Team

This document is an addendum to the CrowdNotifier white paper.[1] The white paper focuses on the high level design and the protocol. This document provides suggestions for implementation and deployment of CrowdNotifier:

- **Entry QR codes.** We unpack the abstract info string and give concrete suggestions for what information should appear in the QR code and how to encode it.
- **Multiple location management.** We show how to simplify the management of tracing information when dealing with many locations. The white paper assumes that each organizer manages a single location, and can thus store the single tracing QR code containing the tracing information. In reality, a single restaurant may consist of several different rooms, or want to tag individual tables. A large company might want to manage many (meeting) rooms at the same time. In these cases, storing tracing QR codes for each location becomes cumbersome. Instead, we show how a single tracing key can be used to manage all rooms and locations under control of the same entity.

1 Entry QR codes

We revisit the components of the entry QR codes. As per the description in the white paper, entry QR codes contain the values $\mathsf{info}, \mathsf{ent}, \pi_{\mathsf{ent}}, \mathsf{notifykey}$. The values ent, π_{ent} and $\mathsf{notifykey}$ are internal to the CrowdNotifier protocol. The goal of the string info is to describe the location.

In practice, the string info should contain *structured data*. The data contained in info must be shown to the user, so that the user can verify that they are indeed checking in to the location described in info. Adding data in a structured format helps apps to present it in a readable format.

We recommend that info encodes the following values:

- *Name.* The name of the location or organizer. For example, "MagiCorp Lausanne".
- *Location.* The geographical location. For example, "Lausanne", or an address. The goal is to uniquely identify a location.
- *Room.* The name of a specific room, when applicable. For example "Upstairs dining room", "AX 11", or "Red meeting room".

[1] see `https://github.com/CrowdNotifier/documents` for the latest version

- *Type.* The type of location or event. For example “bar”, “restaurant”, “lecture room”, or “meeting room”.

Other values, such as a longer description for specific events or a suggested stay time to ease checkout, can be added based on the deployment scenario.

These values should be encoded in a compact structure, for example a protobuf. See the QRCodeContent and QRCodeEntry structures in of our reference implementation as an example.[2]

2 Managing several locations

The approach described in the white paper produces one pair of QR codes for each location: $\mathsf{QR}_{\mathsf{entry}}$ and $\mathsf{QR}_{\mathsf{trace}}$. In this section we explain how to generate several entry QR codes $\mathsf{QR}_{\mathsf{entry}}$ for different rooms, from one secret. This removes the need to manage different values $\mathsf{QR}_{\mathsf{trace}}$ per location.

2.1 A recap of the basic scheme

In the basic scheme described in the white paper the QR code generation takes as input the description of the location info. The generation algorithm then proceeds as follows:

1. It first computes a master public key mpk and two corresponding secret keys: the master secret key msk_L held by the location, and the encrypted secret key $\mathsf{ctxt}_{\mathsf{HA}}$ for the health authority. It sets $\mathsf{ent} = \mathsf{mpk}$.
2. Then it picks random nonces $\mathsf{nonce}_1, \mathsf{nonce}_2$ as well as a notification key $\mathsf{notifykey}$.[3] It sets $\pi_{\mathsf{ent}} = (\mathsf{nonce}_1, \mathsf{nonce}_2)$.
3. Finally, it generates the data for the QR codes. The entry QR code contains info, ent, π_{ent}, and $\mathsf{notifykey}$. The tracing QR code contains $\mathsf{mtr} = (\mathsf{mpk}, \mathsf{msk}_\mathsf{L}, \mathsf{info}, \mathsf{nonce}_1, \mathsf{nonce}_2, \mathsf{ctxt}_{\mathsf{HA}})$ and $\mathsf{notifykey}$.

Note that the tracing QR code largely consists of values that are also available in the entry QR code. Only the values msk_L and $\mathsf{ctxt}_{\mathsf{HA}}$ are private and specific to the tracing process.

To initiate tracing, the venue and the health authorities use their respective master secret keys to compute time-slot and location specific tracing keys. These tracing keys take the form of identity-based decryption keys for the identities

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

[2]Code here: https://github.com/CrowdNotifier/libcrowdnotifier/blob/main/lib/src/v2/messages.proto

[3]Recall that the notification key is used to be able to send encrypted messages that only (notified) visitors can read.

where $\mathsf{hourctr}$ corresponds to the time slot, for example encoded as hours since UNIX epoch.

2.2 Managing multiple locations

In the basic scheme, location owners create one master public key (and corresponding private keys) per location. In this section we show how the same master key can be used for *all locations* managed by a single organizer. Thus drastically simplifying key management.

In the managed scheme, we perform step 1 of the key generation only once to compute the master public key mpk and master secret keys msk_L and $\mathsf{ctxt}_\mathsf{HA}$. We use the master public key mpk for all location managed by the same manager. A local database stores other location/room specific data such as info, nonce_1, nonce_2, and the notification key $\mathsf{notifykey}$.

Given the local database, and the master secret keys, the manager can recover all information needed for tracing, and compute the per location tracing keys. In the following, we describe these steps in more detail below.

The security of tracing information hinges on keeping the master secret msk_L secure. In the following, we describe how this key can be derived from a passphrase. As a result, no security-critical information needs to be stored in the local database. The passphrase itself can be stored in a password management system.

Organization Setup. When creating an organization that manages several locations, the system will generate a strong passphrase of at least 256 bits of entropy. The operator will store the passphrase securely, for example in a password management system. The passphrase is the only security-critical component and is only needed to initiate tracing of rooms. It is not needed to add new rooms to the system.

First, setup now computes the organization's master secret key[4] as

$$\mathsf{msk}_\mathsf{L} \leftarrow H(\mathsf{passphrase}) \mod p,$$

where p is the group order and $\mathsf{mpk}_\mathsf{L} = g_2^{\mathsf{msk}_\mathsf{L}}$ the corresponding owner's public key. In this case the output of H should be much longer than the bit length of p. For example, using SHA512 for p of 256 bits. Alternatively, msk_L can be directly computed by an appropriate method for hashing to the field $\mathbb{Z}_p$ if provided by the cryptographic library.

It then proceeds as in the original QR code generation process to compute the health authority key-pair mpk_HA, msk_HA; the encrypted master secret key $\mathsf{ctxt}_\mathsf{HA}$

[4]For consistency of notation, we will keep writing msk_L and mpk_L subscripted with "L".

for the health authority; and the master public key $\mathsf{mpk} = \mathsf{mpk}_L \cdot \mathsf{mpk}_{HA}$. See the white paper for the details.

The system then stores mpk and ctxt_{HA} in the local database. It does not store any of the other generated values.

Adding a room. To add a room, the operator supplies the information describing the room so that the string info can be computed. See also section 1. The system then proceeds as in the basic scheme, except that it uses the master public key mpk from the database:

1. Retrieve mpk from the local database.
2. Pick random nonce nonce_1, nonce_2 and a notification key $\mathsf{notifykey}$. Let $\pi_{\mathsf{ent}} = (\mathsf{nonce}_1, \mathsf{nonce}_2)$.
3. Generate the entry QR code containing $\mathsf{info}, \mathsf{ent} = \mathsf{mpk}, \pi_{\mathsf{ent}}, \mathsf{notifykey}$.
4. Store a new record in the database for this room containing info, nonce_1, nonce_2, and $\mathsf{notifykey}$.

The entry QR code can be regenerated from the data stored in the database if needed.

Tracing/notifying a room. To initiate tracing, the health authority contacts the operator and specifies the room for which it wants to notify the visitors. The operator uses the passphrase generated initially to recover the information that would normally be in the tracing QR code as follows:

1. Use the provided passphrase $\mathsf{passphrase}$ to recompute the organization's master secret key msk_L.
2. Retrieve the master public key mpk and the encrypted master secret key ctxt_{HA} of the health authority from the database.
3. For the room specified by the health authority, retrieve the values nonce_1, nonce_2, and $\mathsf{notifykey}$ from the local database.
4. Reconstruct the room-specific master trace record $\mathsf{mtr} = (\mathsf{mpk}, \mathsf{msk}_L, \mathsf{info}, \mathsf{nonce}_1, \mathsf{nonce}_2, \mathsf{ctxt}_{HA})$ and proceed as in the white paper to generate time-slot specific pre-tracing keys.