Hashomer – Privacy-Preserving Bluetooth Based Contact Tracing Scheme for Hamagen

Benny Pinkas
Bar-Ilan University, Israel
benny@pinkas.net

Eyal Ronen
Tel Aviv University, Israel
eyal.ronen@cs.tau.ac.il

***Abstract*—In recent months multiple proposals for contact tracing schemes for combating the spread of COVID-19 have been published. Many of those proposals try to implement this functionality in a decentralized and privacy-preserving manner using Bluetooth Low Energy (BLE).**

In this paper, we present "Hashomer", our proposal for a contact tracing scheme tailored for the Israeli Ministry of Health's (MoH) "Hamagen" application. The design is fully decentralized, and has the following properties:

- Message Unlinkability — Different BLE messages sent by the same user cannot be linked to each other (except for messages sent by COVID-19 positive users who *give consent* to tracing their contacts, and only for messages sent within a *short* time period).

- Explainability — To convince users that they were exposed to a COVID-19 positive person, we let them learn the approximate time of contact. This also implies that users can potentially learn, using the phone's GPS information, the location of the exposure.

- Partial Disclosure and Coercion Prevention — Users and the MoH are able to redact tracing information and exposure notifications for specific time intervals.

- Prevention of Relay Attacks – The design prevents attacks where a malicious receiver relays BLE transmissions from one location to other locations.

- Proof of exposure to a COVID-19 positive person — To prevent false reports about exposure, we allow users who are notified by the application about an exposure to a COVID-19 positive person, to prove this fact to the server.

- Identity Commitment — To prevent malicious changing or replacing keys, we bind the BLE messages to a unique ID in a privacy-preserving way.

- Performance — BLE payload size is limited to 16 bytes. The application uses only symmetric key cryptography (AES and HMAC). To reduce bandwidth, contact updates from the MoH are of limited size. Moreover, the local search for exposure is linear in the number of messages and number of COVID-19 positive persons.

I. Introduction

We describe a design for a BLE-based contact tracing application which was consequently developed by the Israeli Ministry of Health (MoH), and deployed in July 2020 under the name Hamagen 2. A reference python implementation of our design can be found at https://github.com/eyalr0/HashomerCryptoRef.

The design is based on two main principles:

- Empowering end-users: The system is decentralized, in the sense that all information is stored locally and is controlled by the users. Only users, and not the government, are informed about exposures to COVID-19 positive persons. The decision on how to behave given this warning is made by the user.
  In order to improve "explainability", namely to convince users to have confidence in the warnings given by the system, it provides users with accurate and fine-grained information about exposures. In addition, to obtain users' trust, the system respects their privacy and enables them to control the information that is revealed if they become COVID-19 positive.
- Preventing attacks: Any attack against the system, or even a publication of the feasibility of such an attack, might reduce public trust and prevent users from using the system or adhering to its exposure notifications. The resulting effect on the public acceptance of the system might be more severe than the immediate results of the attack. We therefore took additional efforts to prevent sophisticated attacks on the trustworthiness of the system. This was done while keeping within the requirements of smooth operation on commodity phones.

The design follows the same structure as other designs for decentralized contact tracing, such as DP-3T [12], Google and Apple [1], and PACT [5]. Namely, each phone sends short BLE messages with random ephemeral ids, and stores similar messages that it receives. When a user is identified as COVID-19 positive, the user can provide the MoH with information that can be broadcasted to all users, allowing them to learn whether they received BLE messages from this user.

A. Privacy, Security and Operational Requirements

We would first like to emphasize several very basic privacy features that this design supports:

1) To guarantee privacy, all information is stored locally. **No messages are ever sent from the application to any server, except for *voluntary* messages by COVID-19 positive persons, or by users who were**

Workshop on Secure IT Technologies against COVID-19 (CoronaDef) 2021
21 February 2021, Virtual
ISBN 1-891562-72-X
https://dx.doi.org/10.14722/coronadef.2021.23011
www.ndss-symposium.org

notified (by the application) about an exposure to another COVID-19 positive user and wishes to report this.

2) The ephemeral IDs sent in short-range BLE messages are computed from keys which are generated in the device, and look pseudo-random to any other entity.
3) The decision whether to reveal contact tracing information is made by the user, and is completely voluntary. Similarly, a notification about exposure to a COVID-19 positive person is shown locally in the application, and is kept hidden from the MoH unless the user wishes to report it.
4) The application does not reveal to the MoH any location information, not even of COVID-19 positive persons.
5) The design *allows the user to "snooze"* sending BLE messages in privacy critical situations (e.g., a journalist meeting with a confidential source). It also *allows users to retroactively delete keys* associated with specific time intervals.
6) The design *allows users to delete their history of exposure alerts*. This feature is needed to prevent other parties, such as employers, from coercing users to prove that they have not received any exposure alerts.

The system must address the following security issues:

1) Prevention of relay attacks, in which an attacker receives messages sent by users of the application in one location, for example, an emergency room in a hospital, and transmits them in different locations (say, busy train stations). Such an attack might cause unnecessary warnings showed to users.
2) Identity commitment — A malicious attacker might try to replace or change keys. For example, a disgruntled employee might want to cause all of his coworkers to self-isolate. To accomplish that, he will bribe a COVID-19 positive person to upload the keys from his phone.
3) Prevention of "fake" exposures – The design prevents users from claiming that they were in contact with an infected person. Some users might opt to do so, for instance, if they want to get a sick leave for the required quarantine period. Note that a user who had a real contact with an infected person can always transfer this information to other users, and then these users will be able to fake a contact. However, the server can become suspicious and do more checks if too many people use the same verification information.
4) Cloning and copying attacks – A sophisticated attacker might clone multiple phones to use the same master key, or copy keys from a phone of an infected person. We do not protect against these attacks.

B. Related Work

A lot of effort has been invested in designing and implementing contact tracing applications. Decentralized contact tracing applications do not keep a central database which records users' contacts, but rather keep this information on users' devices and inform only the users, and no central authority, of contacts with COVID-19 positive persons. Currently the major application that is rolled out across Europe is DP-3T [12]. Another decentralized design, called PACT, was designed by researchers from MIT and other institutes [3]. Centralized contact tracing applications give the government information about exposures. An example of such an application is the TraceTogether application of Singapore [8]. See, for example, [6], [11], for more thorough surveys of contact tracing applications.

Mobile operating system make it hard for applications which run in the background to send and receive BLE communication, and to do this with minimal battery consumption. Apple and Google developed a special API, called GAEN, which supports BLE communication for contact tracing applications. The usage of this API is limited for government applications which agree with usage terms defined by Apple and Google [1].

Security issues with the current designs were identified, e.g. in [13], [4], and demonstrated in [2]. A design which addresses some of these issues was presented in [9] (but not accompanied by an implementation).

There are also contact tracing designs which are based on more advanced cryptographic primitives, such as zero-knowledge proofs of or group signatures, e.g. [7], [14], but these desgins are less efficient, and are not accompanied by any proof-of-concept implementations.

II. Technical Overview

The basic strawman design for a contact tracing scheme has each instance of the application send short random messages (ephemeral IDs) over BLE. The application also records all BLE messages that it receives. During normal operation, the application never sends anything over any other communication channel. The only exception is when the user is identified as COVID-19 positive (and has a code received from the MoH that verifies this fact). Then the user can *explicitly request* to provide the ministry of Health with all the ephemeral IDs that were sent by his copy of the application in the last 14 days.[1] The MoH periodically distributes these values, received from all new COVID-19 positive persons, to all users, where for each time period the MoH sends all corresponding ephemeral IDs in random order. Then any copy of the application can locally check if it previously received any of these ephemeral IDs over BLE, and use a local algorithm to alert the user to the risk of contact with the corona virus (for example, alert the user if ephemeral IDs were consecutively received in any period of 15 minutes.) The user is then able to inform the MoH that he or she were in contact with a COVID-19 positive person.

This initial design provides privacy for all receiving users since no information is sent from their phones, except for short *random* BLE messages. The BLE messages can be generated in each personal copy of the application based on a secret random seed, and look pseudo-random to all other users. The privacy of COVID-19 positive persons is preserved in the sense that it is impossible to link different BLE messages of the same person.

[1] We assume in this paper that epidemiological contact tracing needs to identify all people who were in contact with the COVID-19 positive person in the last 14 days.

This basic scheme has a scalabily issue: Suppose that users change their ephemeral IDs every 5 minutes. Then every user sends more than 4000 ephemeral IDs in the 14 day period. In this case an update message for about 1000 or 10000 new COVID-19 positive persons will be too large. (This is true irrespectively of the compression method that is used, such as a Bloom filter or a Cuckoo filter, as long as we want to keep the false positive probability low.)

Daily seeds: An appealing option is for the client to use a separate seed per day, and use it to derive all ephemeral IDs of that day. In this case the MoH needs to send only 14 seeds per COVID-19 positive person. This approach is currently used by GAEN. However, this approach suffers from some drawbacks: The privacy of the COVID-19 positive persons is affected since it is now possible to link different ephemeral IDs sent by the same person in the same day. The client application needs to generate all ephemeral IDs from the seeds, and this task might be computationally heavy. In addition, it is impossible to redact less than an entire day of the BLE message history of COVID-19 positive persons, namely have the MoH send an update broadcast which does not include some of the ephemeral IDs sent by a COVID-19 positive person. (The redaction feature is important for privacy, efficiency, and accuracy.)

A. The "Hashomer" Design

While the basic idea and design are similar to that of other designs, and in particular DP-3T [12], our design has some privacy and security features which are not present in most other contact tracing designs. We list here the main differences with respect to the DP-3T design.

1) A hybrid design: An initial master key is generated when the application is installed, and a chain of daily master keys is generated from that key. The application keeps the master day key of 14 days ago, and erase all prior keys. Each day is divided to epochs (e.g., hours), and each epoch is divided to time units (e.g., 5 minutes). Every epoch has a different epoch key which is generated from the daily key. The ephemeral ID is changed each time unit, and is derived from the current epoch key. When it is needed to report the ephemeral IDs sent by a COVID-19 positive person, only relevant epoch keys that were used by the person are broadcast to all users.

2) Message Unlinkability: Different BLE messages sent by the same user are pseudo-random and cannot be linked to each other. The goal is to prevent tracking of specific users. The only exception it users who were found to be COVID-19 positive. If such a user agrees to notify their contact, the keys required to identify their messages are uploaded to the MoH servers, and are broadcast to all users. Other applications broadcast keys which enable to link messages sent by the user over relatively long periods of time (a day). Our design has the message-linkability period (epoch) as a parameter, which is currently set to be equal to one hour.

3) Tradeoff Between Privacy and Explainability: The DP-3T design prioritizes the privacy of the positive person and therefore provides to exposed users only a coarse time window of the exposure (e.g., Tuesday morning). We believe that explainability is important in order to convince users to quarantine themselves.[2] Therefore, the application provide users with a more fine-grained time frame about the exposure. Using the device's locally stored GPS history, the application can also provide the user with an approximated location of the exposure.

4) Partial Disclosure: Our design lets the user or MoH redact all tracing information for specific periods of time. This is useful for privacy (for example, letting users redact private meetings which they want to hide), and for efficiency (e.g., redacting times when it is known that the user was alone), and for accuracy (e.g., redacting times where user was stuck in a traffic jam) . To facilitate this, we use a tree structure for key derivation, allowing the MoH to broadcast only a subset of the keys. Although the DP-3T white-paper suggests two variants that support partial disclosures, they require more bandwidth and, as far as we know, are not used in practice.

5) Prevention of Relay Attacks: A potential attack scenario is where an attacker relays messages from one location where COVID-19 positive persons are likely to appear (e.g., an emergency room) to many other busy locations, thus causing many false reports of exposure to positive persons.

To overcome this attack we incorporate the coarse-grain location data of the place of contact into the protocol. The BLE message therefore includes an encryption of coarse geo-location information (e.g., accuracy of ± 500 meters), and a MAC to verify its authenticity. The receiver stores the location in which it received this message. The encryption and authentication keys are only revealed voluntarily by confirmed infected persons. The BLE message *is never sent to the server*, but can be used by the receiver to verify that it was in close physical proximity to the infected sender.

We note that the explainability property requires the receiver to save rather fine-grained timing information, which the receiver can combine with its personal location data. This implicitly enables the receiver to identify locations of contact with COVID-19 positive persons, and therefore the fact that we incorporate encrypted location data in the protocol does not give any new capabilities to the receiver.

The relay prevention feature is optional. If either party (sender or receiver) does not have location information, relay prevention will be disabled for the specific interaction (the relevant party will send or store a NULL symbol as location). As the location information is MACed, malicious relays cannot "remove" location information. As long as a large majority of the users have location information, this mitigation will still be effective.

6) Proof of exposure to a COVID-19 positive person: In some cases, users might falsely report that they were exposed to a COVID-19 positive person, to gain benefits such as priority in COVID-19 testing or a paid sick leave. We, therefore, add an option for users to prove this exposure. This is done by defining part of the ephemeral ID as a function of a verification key. When a user is identified as COVID-19 positive, he or she sends the verification keys to the MoH, but they are not broadcast to all users.

[2]This requirement might depend on the culture and values of different countries. We believe that in many countries, and at least when the system begins rolling out, users will require an explanation for an exposure notification.

Even without these keys, users are still able to identify ephemeral IDs sent by COVID-19 positive persons. They can prove the exposure by providing the part of the ephemeral ID, which depends on the verification key. (The challenge is supporting this feature while keeping the size of the ephemeral IDs to be only 16 bytes)

We note that sending this proof discloses to the MoH's server the specific positive person to whom a user was exposed, so this ability remains optional.

7) Identity Commitment: We want to prevent adversaries from having multiple devices send the same ephemeral IDs. When installing the application, the users commit to a unique ID, that can be an official government-issued ID (e.g., driver license number, social security number, etc.) or a randomly generated number. A computational binding commitment to this ID is used in the BLE messaged generation algorithm. If the user is tested positive for COVID-19, the commitment can be used to prevent a malicious change or replacement of the keys after the commitment.

8) Performance: Our design is limited by multiple real-world operational Constraints:

1) **Short messages:** BLE communication enables each phone to send only relatively short messages. For efficiency and robustness, we require our messages to fit inside the payload of one BLE message, i.e., 16 bytes of data.[3]
2) **Efficiency:** To reduce the performance and battery requirements of our solution, the application must use limited CPU resources. We therefore limit it to using only efficient symmetric key cryptography (AES and HMAC). Moreover, real time calculations, such as computing the ephemeral ID, use only AES rather than HMAC.
3) **Limited download channel:** A server run by the MoH will publish the information that is required to check for exposure to new COVID-19 infected people. Each copy of the application must download this data, but we can only expect users to download a few megabytes of data per day. We limit the size of the data to approximately 5 MBytes for 1000 new COVID-19 positive persons per day. Our design supports *dynamic* trade-off between the length of the linkabilty period and size of the contact updates. Increasing this period from one hour to one day allows us to support 24000 new COVID-19 positive persons per day with the same 5 MBytes size.
4) **Run time of exposure detection algorithm:**The complexity of the algorithm that checks for exposure is linear in the number of BLE messages received and number of COVID-19 positive persons.

III. Cryptographic Design

In this section we will describe some central technical points in our cryptogarphic design. The full version with the detailed cryptographic design can be found in the full specification of the system [10].

[3]Some of the other projects use the same message length, for example the Privacy-Preserving Contact Tracing project of Apple and Google.

A. Key Derivation

We divide the time to days, and each day to epochs of T_E minutes. We suggest setting $T_E = 60$ and therefore a day has 24 epochs. We further divide the time to units of T_U minutes, and suggest using $T_U = 5$. Each device will change the ephemeral ID that it sends every T_U minutes. (The reason for changing the ephemeral ID more frequently than changing the epoch is to prevent linkage between different ephemeral IDs that a user sends in the same epoch, unless the user is later diagnosed as COVID-19 positive.)

We use a tree-like key derivation scheme. The goal is to allow the server to broadcast separate keys for each epoch. This makes it much harder to link messages of the same infected person across different epochs. However, the scheme also allows the server to broadcast daily keys for reducing the total bandwidth.

We chose to derive all keys in two steps, first deriving daily keys from the master key, and from these daily keys deriving epoch keys. This allows the server to store the daily or epoch keys in a lexicographic order, without information that allows to link contacts across different days or epochs. Note that from a privacy perspective it is always preferable to only store epoch keys. However, we support the option of storing and sending daily keys in case that the of number of infected persons is large and it is required to optimizing the bandwidth.

The key derivation tree is depicted in Figure 1. Due to the page limit we do not provide here the full description of the key derivation process, and refer the user to the full version of this paper or to our cryptographic specification in [10].

Design Intuition: The key $PreK^{i,j}_{\text{epoch}}$ is derived from the day key. The key $K^{i,j}_{\text{epoch}}$ is derived from $PreK^{i,j}_{\text{epoch}}$ using the key K^{i}_{com} which binds the keys to a unique ID value. On the other hand, we do not want to require the server to broadcast K^{i}_{com} to users since this value will enable them to connect values sent by an infected person in different epochs. Therefore we derive $K^{i,j}_{\text{epoch}}$ from $PreK^{i,j}_{\text{epoch}}$, and have the server only broadcast $K^{i,j}_{\text{epoch}}$ to users. The keys $K^{i,j}_{\text{epochENC}}$ and $K^{i,j}_{\text{epochMAC}}$ are derived from $K^{i,j}_{\text{epoch}}$. These keys will be used by users to verify that the BLE message they received corresponds to a infected person. The key $K^{i,j}_{\text{epochVER}}$ will be used to ensure that only users which receive the ephemeral ID can reconstruct a "proof" which depends on this key. This value verifies to the server that the user indeed received this BLE message from a COVID-19 positive person.

B. Ephemeral IDs

We first list the rational for the design that we suggest:

1) The ephemeral ID should include an authenticated encryption of the location of the transmitting device. This is required in order to prevent relay attacks that broadcast the ephemeral ID to users in other locations. The design must also prevent attacks where the relay changes the original ephemeral ID, for example by xoring values to the ciphertext in order to change the original encrypted location to another location.

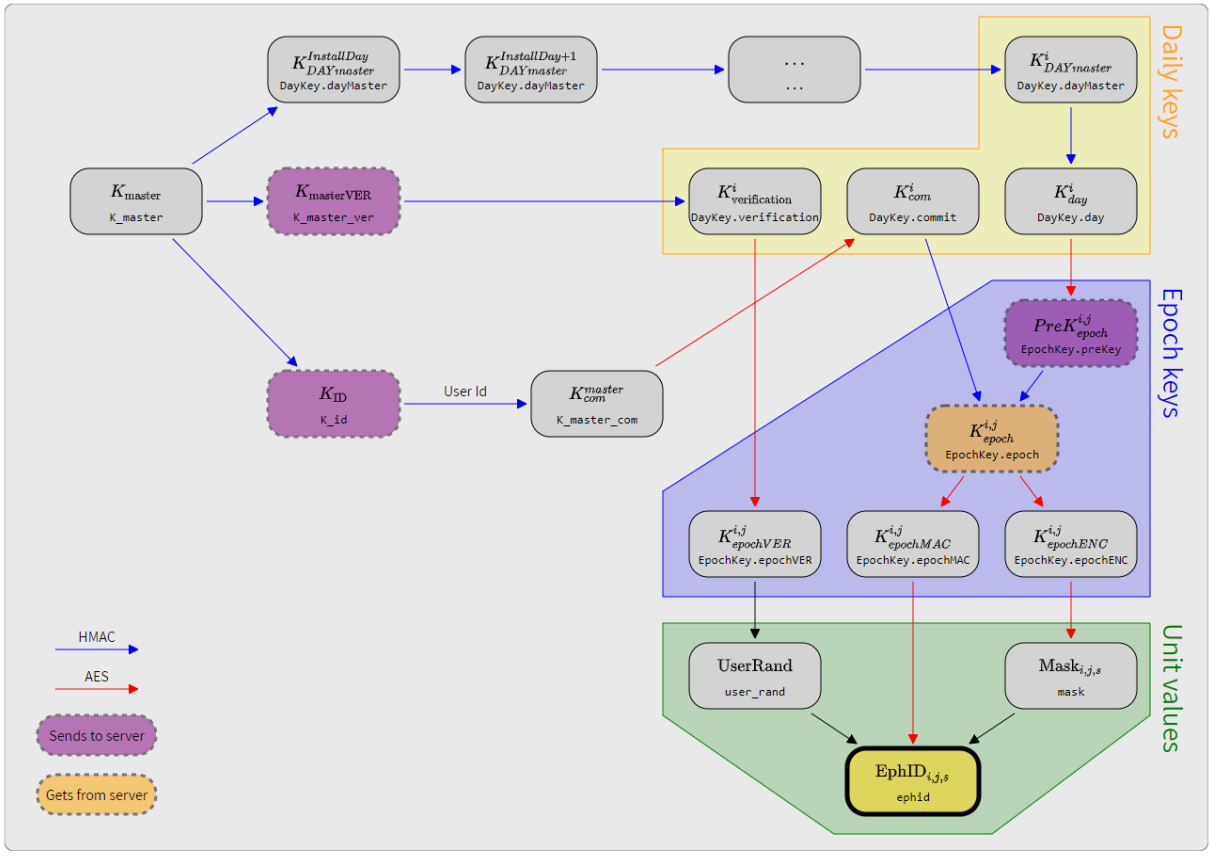

Fig. 1: The key derivation mechanism.

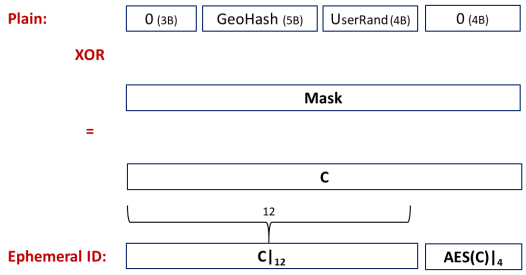

Fig. 2: The derivation of the Ephemeral ID.

2) There is a constraint of sending only 16 bytes in the ephemeral ID.
3) Another constraint is that the computation must be quick, since it must be run in real time (unlike the key derivations). Therefore we prefer using AES to using HMAC.
4) Checking at the client side for physical contacts, should be run in time which is linear in the number of infected persons and the number of recorded beacons.

The derivation of the Ephemeral ID described in Figure 2. The ephemeral ID for time unit s in epoch j of day i is a 16 byte value that is generated in the following method. The application sets a random 4 byte value, UserRand, as a truncation of the epoch verification key $K^{i,j}_{\text{epochVER}}$. It also computes a 5 byte GeoHash of its current location. It then runs the following computation.

$$\text{Mask}_{\text{i,j,s}} = \text{AES}(\text{K}^{\text{i,j}}_{\text{epochENC}}, \text{s})$$
$$\text{UserRand} = \text{Truncate}(\text{K}^{\text{i,j}}_{\text{epochVER}}, 4)$$
$$\text{Plain}_{\text{i,j,s}} = 0\ (3\text{B}) \,||\, \text{GeoHash}\ (5\text{B}) \,||\, \text{UserRand}\ (4\text{B}) \,||\, 0\ (4\text{B})$$
$$\text{C}_{\text{i,j,s}} = \text{Plain}_{\text{i,j,s}} \oplus \text{Mask}_{\text{i,j,s}}$$
$$\text{EphID}_{\text{i,j,s}} = \text{Truncate}(\text{C}_{\text{i,j,s}}, 12) \,||\, \text{Truncate}(\text{AES}(\text{K}^{\text{i,j}}_{\text{epochMAC}}, \text{C}_{\text{i,j,s}}), 4)$$

A note on UserRand: The UserRand field is used to enable the receiver of the ephemeral ID to prove that it received this message. Ideally, this message would be unique per ephemeral ID, for example computed as $\text{Truncate}(\text{AES}(\text{K}^{\text{i,j}}_{\text{epochVER}}, \text{s}),\ 4)$. However, since the ephemeral ID is generated in real time, we prefer to improve the latency of this operation and use the same UserRand value for all ephemeral IDs sent in an epoch. This will enable the receiver to prove that it received an ephemeral ID in this epoch, but not the exact time and number of ephemeral IDs received.

MAC size: Due to communication constraints, our MAC

is only 32-bits long. However, we assume that for deciding on exposure, the application will require at least two valid ephemeral IDs in a short time interval. And that means that in most cases, a relay attacker needs to forge two or more messages, and that will happen with a negligible probability of approximately 2^{-64}.

A note on GPS information accuracy and availability: We propose sending only coarse geo-location information (e.g., accuracy of ± 500 meters). We believe this is sufficient to prevent most cases of relay attacks, without compromising the users' privacy (Bluetooth range is much less than 500 meters). As location information might not be available (e.g., a long tunnel, location is disabled by the user, etc.), the information encoding should allow notifying that no location information is available (e.g., all zero bytes).

A note on the storing the GPS information: The Geo-Hash, which encodes the location, need not be kept secret from the receiver of the BLE message, since the receiver is physically close to the sender. Moreover, we assume that the phone may record its own GPS information. However, one can be worried that external access to the data stored on the device (for example by compromising the device, or using a court order), will reveal the location data. It would have therefore been preferable for the device to store a MAC of the geo-location, keyed by a key which is sent in the ephemeral ID and deleted afterwards. However, the current length of the ephemeral ID does not support a solution of this type.

C. Communication with the Ministry of Health and Checking for Exposure

For full details about the process of uploading keys to the MoH by COVID-19 positive persons, key broadcast, and local check for exposure see the full specification of the system [10].

IV. Deployment

A BLE-based contact tracing application, Hamagen 2, was deployed by the Israel Ministry of Health in late July (see source code at https://github.com/MohGovIL/rn-contact-tracing). This application was an update to a previous version of the application, Hamagen 1, which was based on GPS location data. Hamagen 1 initially had a very large number of downloads, but as time passed a large percentage of its users stopped using it. Unfortunately, the deployment of Hamagen 2 was a failure, and very few people chose to use it on a regular basis. We can identify two major reasons for this failure: technical issues, and trust issues.

a) Technical issues: The application suffered from various usability issues, especially in the first versions after the initial release. The issues included high battery consumption on some phones, and lack of support for older and some new phones. Bluetooth connection with other devices was also affected in some cases. These issues arose from two main reasons:

1) The release date was determined by a government resolution, without providing time for beta testing. The application was only briefly tested by a small group of developers on about 30 devices. The next step was releasing the application to more than a million users.
2) Apple and Google prevented the MoH from using the GAEN API. The application developers had to bypass the various limitation on BLE-based contact tracing without the OS support. The Israeli MoH decided that location data is needed in order to support location-based tracing of contacts with people who do not use smartphones. (In Israel, a large percentage of the population does not use smartphones for religious reasons, and the GAEN API does not provide a solution for them.) Although this unique local issue was explained to Apple and Google, they did not agree to change their global restriction on using GPS location together with the GAEN API.

b) Trust issues: Even taking into consideration the technical difficulties, we argue that the main reason for the failure of the deployment was the lack of trust among the general public. The implementation followed our design and was endorsed by prominent privacy advocates, but this fact was not communicated to the public. Moreover, since the beginning of the pandemic, Israel has been trying to trace contacts using intrusive surveillance technology of the Israel Security Agency. Without adequate publicity, which was lacking, the public was unable to distinguish between that type of surveillance, and the privacy-preserving design of Hamagen 2. In addition, people also had a general distrust for the government's response to COVID-19. As a result, people were reluctant to install any application that, in their view, might be spying on them.

While we lack professional expertise in this area, we believe that if the public had trusted the effectiveness and privacy of the contact tracing solution, many people would have used it despite the technical difficulties. Therefore, trust issues were the primary reason for the failure of the application deployment.

Acknowledgment

We thank the many researchers with whom we discussed the design, including Manuel Barbosa, Eli Biham, Dan Boneh, Yehuda Lindell, Moni Naor, and Adi Shamir. We are very grateful for Ron Asherov and other contributors for their hard work on the Python reference implementation and helpful comments.

This work was supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, by the Alter Family Foundation, and by Len Blavatnik and the Blavatnik Family foundation. Eyal Ronen is a member of CPIIS.

References

[1] Apple and Google. Exposure notification cryptography specification, April 2020. https://www.apple.com/covid19/contacttracing/.

[2] Lars Baumgärtner, Alexandra Dmitrienko, Bernd Freisleben, Alexander Gruler, Jonas Höchst, Joshua Kühlberg, Mira Mezini, Markus Miettinen, Anel Muhamedagic, Thien Duc Nguyen, Alvar Penning, Dermot Frederik Pustelnik, Filipp Roos, Ahmad-Reza Sadeghi, Michael Schwarz, and Christian Uhl. Mind the GAP: security & privacy risks of contact tracing apps. *CoRR*, abs/2006.05914, 2020.

[3] 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. Cryptology ePrint Archive, Report 2020/863, 2020. https://eprint.iacr.org/2020/863.

[4] Paul-Olivier Dehaye and Joel Reardon. Swisscovid: a critical analysis of risk assessment by swiss authorities, 2020.

[5] Gary F. Hatke, Monica Montanari, Swaroop Appadwedula, Michael Wentz, John Meklenburg, Louise Ivers, Jennifer Watson, and Paul Fiore. Using bluetooth low energy (BLE) signal strength estimation to facilitate contact tracing for COVID-19, 2020.

[6] Archanaa S. Krishnan, Yaling Yang, and Patrick Schaumont. Risk and architecture factors in digital exposure notification. Cryptology ePrint Archive, Report 2020/582, 2020. https://eprint.iacr.org/2020/582.

[7] Joseph K. Liu, Man Ho Au, Tsz Hon Yuen, Cong Zuo, Jiawei Wang, Amin Sakzad, Xiapu Luo, and Li Li. Privacy-preserving covid-19 contact tracing app: A zero-knowledge proof approach. Cryptology ePrint Archive, Report 2020/528, 2020. https://eprint.iacr.org/2020/528.

[8] Singapore Ministry of Health and Government Technology Agency. TraceTogether, 2020. https://www.tracetogether.gov.sg/.

[9] Krzysztof Pietrzak. Delayed authentication: Preventing replay and relay attacks in private contact tracing. *IACR Cryptol. ePrint Arch.*, 2020:418, 2020.

[10] Benny Pinkas and Eyal Ronen. Hashomer – a proposal for a privacy-preserving bluetooth based contact tracing scheme for Hamagen, 2020. https://github.com/eyalr0/HashomerCryptoRef/blob/master/documents/hashomer.pdf.

[11] Leonie Reichert, Samuel Brack, and Björn Scheuermann. A survey of automatic contact tracing approaches. Cryptology ePrint Archive, Report 2020/672, 2020. https://eprint.iacr.org/2020/672.

[12] 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.

[13] Serge Vaudenay. Centralized or decentralized? the contact tracing dilemma. Cryptology ePrint Archive, Report 2020/531, 2020. https://eprint.iacr.org/2020/531.

[14] Zhiguo Wan and Xiaotong Liu. Contactchaser: A simple yet effective contact tracing scheme with strong privacy. Cryptology ePrint Archive, Report 2020/630, 2020. https://eprint.iacr.org/2020/630.