DENSE: A Decentralized Exposure Notification Spreader for Densely Populated Settings

Ruslan AlJabari
Nathan Bhak
Jerry Chen
Ryan Guan
Stanford University

ABSTRACT

DENSE is a mobile-phone-based contact tracing protocol that uses a fully decentralized architecture (rare among existing contact-tracing systems) and preserves privacy. Devices in close proximity pair with one another using Bluetooth Low Energy, logging their recent encounters and forwarding infectious disease notifications to other nearby devices. We hypothesize that this system would work effectively in densely populated settings like college campuses, where most in-person interactions will occur in similar locations.

1 INTRODUCTION

Contact tracing and exposure notification systems are important public health measures in the face of widespread infectious diseases. In order to quickly notify people who may have come into contact with someone infected, most of these systems use a central server or set of servers to store user data, which can lead to concerns about user privacy. With DENSE, we explore a *completely decentralized* model for transmitting COVID-19 exposure notifications. Instead of communicating with a central server, in DENSE's peer-to-peer/gossiping protocol, users continuously advertise COVID-positive notifications they receive from nearby users. If the network of users is densely connected enough, eventually any user will receive a notification from any other user. We hypothesize that this model would be successful in densely populated settings such as college campuses, where people are frequently in close proximity to each other, and where COVID-positive people may remain near COVID-negative people after testing positive, and we propose that this model will function even as people join and leave the network at arbitrary times. We implemented the DENSE protocol for iOS and evaluated its performance on a small network.

2 RELATED WORK

DENSE takes inspiration from various contact tracing protocols implemented and used during the COVID-19 pandemic.

The PEPP-PT and BlueTrace protocols involve communicating a user's full encounter logs to a central server, owned by a health authority, if they have tested positive. The health authority server can decrypt information in the logs to reveal each user's server-issued identifier (for PEPP-PT) or unique static ID and phone number (for BlueTrace).[2] [3] These systems send full logs of encounters to a central server, even a trusted one, raising privacy concerns. DENSE uses some similar ideas to PEPP-PT to preserve user privacy: using random and changing identifiers, storing most encounter logs on-device to avoid data leaks, and periodically clearing encounter logs.

The DP-3T protocol is a protocol designed with data privacy as a first consideration. [7] Although it still utilizes a central server to relay messages, DP-3T only trusts the server to not tamper with exposure events (i.e. not creating fake or removing real exposures) but notably does not trust the server with user privacy. Rather than upload a full log of encounter, positive users will only send a representation of their own ephemeral identifier that the server will relay to other devices, which

reconstruct the real ephemeral ID offline. A similar protocol is implemented by Apple and Google in their mobile devices. [1] Even though the server does not process logs in these protocols, a central health authority still carries the risk of observance of message relays; an attacker that can assume control of the central server can track individuals based on the identifiers that are observed.

While these protocols notify users if they have been potentially exposed to a positive case, another contact tracing system known as NOVID alerts users to whether they came in contact with a positive case or were in close proximity to someone who themselves were at some point in contact with a positive case. [6] By determining degrees between users in their physical interaction networks, NOVID aims to notify individuals about exposure risk before exposure occurs. To facilitate this, a central server stores information about individuals' interaction networks, requiring additional trust in the server to protect private user data.

DENSE's fully decentralized model is likely less efficient at transmitting exposure notification messages than the aforementioned centralized systems. Even with a dense network (e.g. a single college dorm rather than a whole college campus), messages may take several peer-to-peer hops to reach all intended recipients. A longer identifier lifetime than existing systems also results in more possible vulnerability to eavesdropping and tracking. However, this decentralized model for exposure notification eliminates any concerns from client communication with and data processing on a central server, in addition to being theoretically interesting.

3 DENSE NETWORK STRUCTURE AND PROTOCOL

DENSE users within 100 meters of each other are able to exchange messages using Bluetooth Low Energy, a wireless communication medium that uses little power.[4] Periodically, while the app is open, users scan for other users using the same service; at all other times, users' devices advertise their presence on Bluetooth. Once a connection has been established, messages can be read from and written to between devices.

These messages can be categorized into two types. First, users exchange RSA public keys with other users and record these encounters, in order to determine whose infection notifications they should pay attention to. Second, if a user knows they have COVID, they can broadcast an infection notification message including their own key as an identifier. Only users who had previously come into contact with the infected user, i.e. those who needs to know about someone's infection, will be able to decrypt and verify this message. Bystanders who cannot decrypt infection notifications still forward them through the network. Then, in a densely connected network, these messages should reach all intended recipients.

In this section, we describe these messages' structure, the maintenance of associated logs, the process of rotating keys/identifiers, and some properties of the protocol overall.

3.1 Pairing messages for encounter logging

3.1.1 Description of protocol. A user generates a 4096-bit RSA key pair upon entering the DENSE system. In order to exchange public keys, users broadcast their unencrypted DENSE public key through a value on the device readable to anyone who can open a Bluetooth connection with them. Users broadcast their keys rather than a pseudonymous identifier because the DENSE system lacks a centralized infrastructure to map keys with identifiers. Instead, in DENSE, public keys serve as the temporary pseudonymous identifiers used by other exposure notification systems.

When a user discovers a DENSE key broadcast, they record the reported public key, mapped to the time they received the message, in their encounter log. This process of logging constitutes a "pairing". If the received public key is already in the user's encounter log, the time associated with it will be overwritten with the time the new message was received; for infection notification purposes, only the most recent encounter time is necessary.

To reduce memory footprint, the encounter log is routinely "garbage-collected". Each day, the encounter log is scanned and entries more than 14 days old are cleared automatically. This constant is similar to that in related work, because an encounter from over 2 weeks ago is likely no longer relevant for the purposes of contact tracing.

To preserve privacy, keys/identifiers are rotated periodically. Once a user has been advertising one public

key for at least 7 days, they generate a new key pair and begin to share the new public key. Incoming messages for the next 7 days should be decrypted with either the previous private key or new private key, so that any encounters from before the key rotation are adequately captured.

Finally, we do not include responses or acknowledgment messages in this part of the protocol because a Bluetooth connection already requires a two-sided agreement on the connection.

3.2 Infection notification messages

3.2.1 Description of protocol for senders. A user who has tested positive for COVID periodically advertises DENSE *infection notification messages* through a distinct value on device. Each message contains an unencrypted header identifying it as an infection notification and including the time that the message was sent. The remaining content — the public key of the infected user in addition to a duplicate of the time — is symmetrically encrypted with a randomly generated secret key. The secret key is attached as multiple copies encrypted with each of the RSA public keys listed in the infected user's encounter logs at the time of sending. Lastly, the encrypted content is digitally signed with the infected user's RSA private key using RSASSA-PSS. The characteristic value that advertises the infection notification is the same no matter if the user was originally sending or forwarding the message. Then, receiving users should be able to find out who was infected only by decrypting and verifying the message, and a receiving user can do this iff their public key was in the infected user's encounter logs.[5]

3.2.2 Description of protocol for receivers. Receivers of relevant infection notification messages will broadcast the message to nearby users through the same characteristic value, and will notify the user if the infection notification message is intended for them.

First, steps are taken to prevent forwarding fraudulent messages. If the time in the header is over a week ago, the message is discarded as irrelevant to contact tracing. Next, the receiver attempts to decrypt the body of the message using their own private key. If the decryption is successful, the message was intended for this receiver. However, if the body's time does not match the header time, or the signature is not verified by the public key included in the body, the message is discarded as it was tampered with. If not, the message is a legitimate notification of infection: DENSE should alert the user along with the last time they were in contact with the infected person, if the public key exists in their encounter logs.

Finally, if the message was not discarded from the above steps, the receiver stores the encrypted message in a received infection notifications log, so that it can be broadcasted unchanged to other nearby users. As in the encounter log, we periodically compare each notification's timestamp with a standardized time-to-live (e.g. 7 days), removing it from the log if necessary.

3.2.3 Discussion. If keys function theoretically as pseudonymous identifiers, why take the extra step of encrypting and signing the infected user's public key? This step is taken for two reasons. First, although public keys are pseudonymous identifiers, bystanders ought not to know the public key of someone infected if they have not been in contact. Second, this can prevent pseudo-replay attacks where a malicious user edits the timestamp on a stale request to fool legitimate users into thinking that they have had a recent COVID-positive encounter with another legitimate user who actually tested positive for COVID in the past.

3.3 Properties of DENSE

Desirable theoretical properties of a contact-tracing system include privacy preservation, robustness in the face of malicious users (such as those seeking to forge other users' messages), and actual effectiveness at communicating exposure notifications. We discuss the first two.

3.3.1 Privacy. One issue with any system that tracks users' encounters or broadcasts information about users is compromising user privacy. DENSE attempts to evade this issue by avoiding the concentration of data in any central server, clearing logged data frequently, and using only pseudonymous, temporary identifiers.

While the information stored by the app itself only includes the most recent encounter time, a malicious user who can intercept Bluetooth messages will be able to determine, for example, which users (public keys) visited a certain area at which times. While Bluetooth connections are established 1:1, legitimate users will actively seek out Bluetooth connections to share their keys and

any other messages, so a malicious user would not even have to finagle becoming a person-in-the-middle to find a legitimate user's key.

However, we believe that this situation does not present a privacy/tracking issue because users periodically rotate the key they advertise. At worst, a single user's general location could be identified for no more than a week, if the malicious user was able to follow them around or plant Bluetooth devices in multiple locations. In addition, this user would not be easily identifiable, because users share no personally identifying information through DENSE, including exact GPS information, phone number, or name.

3.3.2 Malicious user tolerance. In the previous section, we considered a malicious eavesdropper on DENSE messages. Now we consider an attacker who sends DENSE messages with the aim of disrupting the system.

Because there is no way of authenticating a public key broadcast, an attacker could share a public key that does not belong to them. However, a legitimate user listening will do nothing but log the key, and will only send a message in response if they report an infection. If a legitimate listener did send an infection notification, it would not be decipherable by the attacker, and they would already sent a message anyway, so the attacker's actions could not result in a legitimate user sending additional messages.

A malicious user might also attempt to forge an infection notification message with a fraudulent public key, but this message will not cause an alert for anyone who hasn't logged this key. Replicating a legitimate public key message will cause it to be discarded if the timestamp was altered or more than 7 days old.

In sum, existing attack vectors are to overwhelm someone's logs with key advertisements from multiple fraudulent public keys, or overwhelm the network with repeated non-duplicate infection notification messages.[1] With no client-server communications — only client-client communications where we must trust on first use — we believe there are limited ways to guard against these network-flooding attacks.

[1]In this latter case, a user who receives a false alert will likely take steps to protect their own health. We think this would actually be a good outcome in the middle of a public health crisis that necessitates contact tracing.

4 IMPLEMENTATION AND EVALUATION

We implemented the core message-generating and message-passing features of DENSE for iOS using React Native, relying on the `hybrid-crypto-js` library for cryptographic primitives.

To describe how we evaluated the system, we must describe the constraints of Bluetooth Low Energy. Bluetooth in general categorizes devices into "centrals", which scan their surroundings for other devices, and "peripherals", which advertise their presence. (Phones are able to serve as either, allowing for the shifting between roles in our description of the protocol.) Each device advertises certain "services" — bundles of data storage and features. Services operate using multiple "characteristics", which are readable and writeable values along with security properties and configuration information. For example, BlueTrace has established its own service for communicating information between phones. [3] [4]

Ideally, to implement DENSE over Bluetooth, we would create a unique BLE service for DENSE. Messaging would proceed by phones switching between central and peripheral roles and reading from/writing to each other's characteristics in order to communicate, as described.

However, we could not find a publicly available and free way to allow a phone to advertise a new service, and the phones we used to test DENSE advertised existing services that had single characteristics that were either read-only or write-only. As a result, we could not test DENSE using only our phones. We also could not use emulators because Bluetooth capabilities are not enabled. In order to evaluate and demo the system, our only feasible option was to set up a laptop as a Bluetooth peripheral advertising a unique service, and communicate between phones by having each phone write to characteristics from this service on the laptop, rather than to the other phone.

In this setting, we found that two phones acting as Bluetooth central devices could successfully communicate pairing messages and infection notifications. Because we used the laptop only to mediate Bluetooth communications, we believe this demonstration shows that our methods are sound, in allowing messages to be communicated and decrypted correctly. Unfortunately, however, we did not think that this format would be

suitable for a larger-scale implementation or evaluation, because DENSE was designed to be a decentralized, phone-to-phone protocol and it would be irrelevant to evaluate the effect of network conditions or a larger network when a laptop would have to be used as a central intermediary.

5 CONCLUSIONS AND FUTURE WORK

DENSE allows users to transmit COVID exposure notifications with no central server or authority involved. Even though the lack of a central server necessitates that users share more with each other over the network, we believe user privacy is still achieved — even in the face of a malicious user able to intercept and send messages — by frequently rotating publicly transmitted keys and not distributing personally identifying information.

One future design idea that would allow for more user choice is to enable users to choose privacy levels of an infection notification. In order to share more details about their exposure to an infected individual, such as the location of the encounter, we could implement mechanisms to mark certain contacts as “trusted", allowing them to decrypt and view more information from the exposure message than others that receive the notification. The tradeoff is that more personally identifying information must be revealed — both to determine whether to trust another user and to provide more information about the encounter.

Most importantly, we did not undertake larger-scale evaluations of DENSE in real settings due to the difficulties mentioned in the evaluation section. As a result, we do not know whether the network-flooding protocol is sufficient to enable tracing at scale, nor if the conditions needed to share keys sufficiently approximates conditions that would result in a close contact. In addition, DENSE only uses Bluetooth, like similar digital contact tracing protocols. We did not explore transmitting messages or logging encounters via other mediums, such as identifying nearby users by their connected Wi-Fi hotspot. A more elaborate implementation and further evaluation would demonstrate additional room for growth.

ACKNOWLEDGMENTS

We would like to thank CS244B course staff, Professor David Mazières and Geet Sethi, for providing feedback on our project during its development.

REFERENCES

[1] 2020. Exposure Notification Bluetooth Specification. https://www.apple.com/covid19/contacttracing

[2] 2020. Pan-European Privacy-Preserving Proximity Tracing: High-Level Overview. (2020). https://github.com/pepp-pt/pepp-pt-documentation/blob/master/PEPP-PT-high-level-overview.pdf

[3] Jason Bay, Joel Kek, Alvin Tan, Chai Sheng Hau, Lai Yongquan, Janice Tan, and Tang Anh Quy. 2020. BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders. *Government Technology Agency-Singapore, Tech. Rep* 18 (2020). https://bluetrace.io/static/bluetrace_whitepaper-938063656596c104632def383eb33b3c.pdf

[4] Robin Heydon and Nick Hunn. 2012. Bluetooth Low Energy. *CSR Presentation, Bluetooth SIG* (2012). https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx

[5] IEEE. 2000. IEEE Standard Specifications for Public-Key Cryptography. *IEEE Std 1363-2000* (2000), 1–228. https://doi.org/10.1109/IEEESTD.2000.92292

[6] Po-Shen Loh. 2020. Flipping the Perspective in Contact Tracing. https://doi.org/10.48550/ARXIV.2010.03806

[7] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth Paterson, Srdjan Čapkun, David Basin, Jan Beutel, Dennis Jackson, Marc Roeschlin, Patrick Leu, Bart Preneel, Nigel Smart, Aysajan Abidin, Seda 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. 2020. Decentralized Privacy-Preserving Proximity Tracing. arXiv:2005.12273 [cs.CR]