Centralized or Decentralized?

The Contact Tracing Dilemma

Author: Serge Vaudenay

Presenter: Chaitanya Rahalkar

What is this paper all about?

- This paper studies, debates and compares on the effectiveness of several centralized and decentralized contact tracing solutions that were introduced during the COVID-19 pandemic.

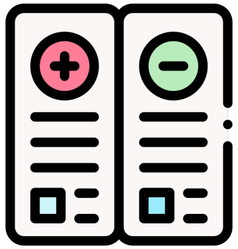
- The authors talk about the vulnerabilities, privacy violating attacks and advantages of both centralized, and decentralized contact tracing.

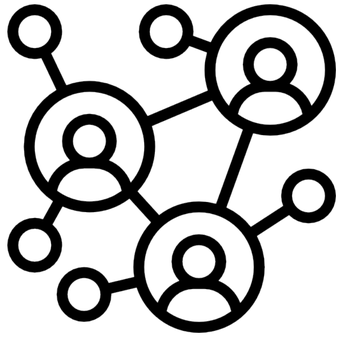
“Contact tracing should be developed as fast as possible but as slow as necessary” - Alain Berset

Which one is better?

🚫 **None!**

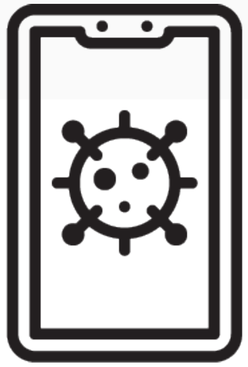
Both have their merits and demerits!

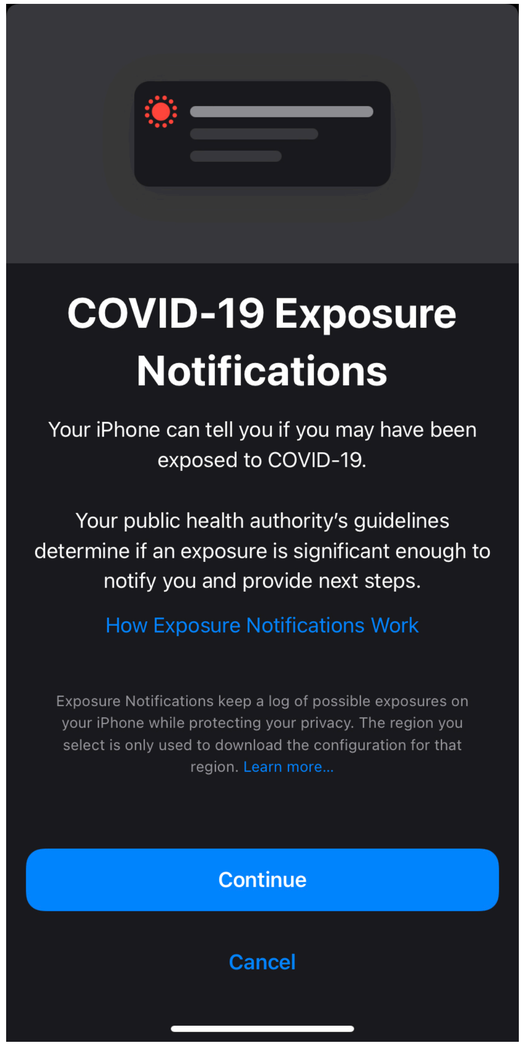




Introduction

Contact Tracing: A Quick Refresher





Introduction

Requirements of Good Contact Tracing Solutions

- Ensure privacy of millions of people as much as possible
- Making clear statements about the implications of the tools using these algorithms.
- The algorithm should not be used for other purposes (including the abuse of the app by both users and authorities)
- The system should minimize the process over private data and make sure no participant could learn more than what he / she already knows.

Introduction

Brief History of Contact Tracing

- PEPP-PT (Pan-European Privacy-Preserving Proximity Tracing), offered both centralized and decentralized solutions. Decentralized was called DP3T. DP3T team criticized centralized contact tracing solutions.

- However, this was just criticism. No academic discussion and debate took place (one of the reasons for this paper).

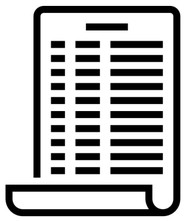
- ROBERT and NTK (centralized solutions) released their specifications upon pressure by DP3T group.

- Several experts debated on whether both the solutions are really privacy-preserving. Paradoxical to call “contact tracing” system as “privacy preserving”.

Introduction

Contact Tracing Solutions Studied

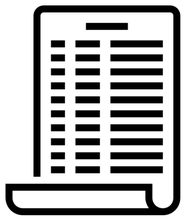
- Centralized - TraceTogether, ROBERT and NTK
- Decentralized - DP3T, Canetti-Trachtenberg-Varia, PACT-East, PACT-West, TCN Coalition and the Apple-Google solution.



Introduction

Joint Statement Proposed

- Hundreds of researchers signed a joint statement for contact tracing solutions to satisfy four imperative requirements -
  1. Apps must only be used to support public health measures for the containment of COVID-19. It shouldn’t be used for any purpose other than this.
  2. Any considered solution must be fully transparent, including its protocol and app implementations (open-sourced)
  3. When multiple options exist, the most privacy-preserving one should be chosen.
  4. The use of the contact tracing apps and the systems that support them must be voluntary, used with the explicit consent of the user. Systems must be able to delete the data and switched off when COVID-19 is over.



Introduction

Joint Statement Proposed

This point and some additional sections of the letter were used by advocates of decentralized systems as an endorsement by academia. The letter states that centralized solutions reveal social graph. This is exaggerated.

3. When multiple options exist, the most privacy-preserving one should be chosen.

Introduction

Factors Considered During Analysis and Debate

1. Tracking People - Risk of depseudonymization of ephemeral identifiers (linking identifiers and coercion attacks)

2. Disclosing the Social Graph - Debunk the exaggeration that only centralized systems reveal the graphs.

3. Identifying diagnosed people - Seen mostly in decentralized solutions. Sybil attacks in centralized systems (creating dummy accounts)

4. Pressure to opt in - Mindsets of privacy-freaks and grumpy users

5. Injecting false at-risk alerts - Replay / Relay attacks, bribing diagnosed people to submit false reports.

Overview of Contact Tracing Systems

- In all systems, $app_{U,}$ is loaded with a list L_U^{out}, of ephemeral identifiers $(e_1,\ldots,e_n)$.
- Each identifier is released during its *epoch.* During an epoch, app_U instructs the phone to broadcast e_i.
- At the same time, app_U collects the ephemeral identifiers e_j' which are received from app_V.
- Hence, there are two lists with app_U - L_U^{out} and L_U^{in}.

Overview of Contact Tracing Systems

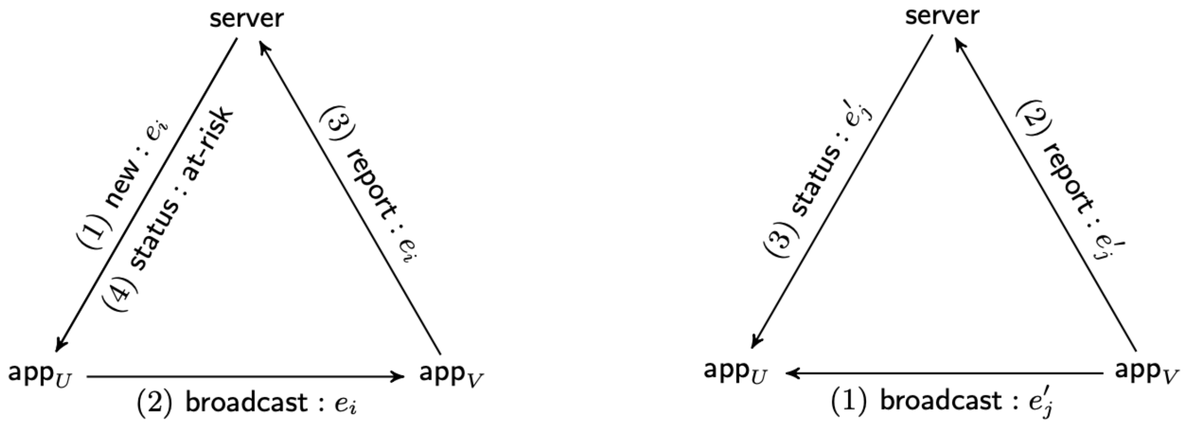


Fig. 1. Information flow in centralized (left) and decentralized (right) contact tracing when V, holder of app_V, is diagnosed

L_U^{out} is obtained from server

L_U^{out} is generated by app_U

Overview of Contact Tracing Systems

Working of Centralized System

- Registration: Each app registers with the server. Server sets up a pseudonym for the app to map the user.

- Setup of new identifiers: App contacts server to setup new identifiers. The server creates a list of e_i which are mapped to pseudonyms using a trapdoor (we assume that the server has a trapdoor).

- Broadcast: During each epoch, the app broadcasts e_i. The app on other phones collects these broadcasted identifiers and persists them.

- Reporting: Upon positive diagnosis, user provides app with the appropriate credential to upload to the server.

- Status Verification: App connects and authenticates to server to check status of the user on the server. Server answers whether the user is at risk or no.

Overview of Contact Tracing Systems

Working of Decentralized System

- Setup of protocols: App prepares a list of random e_i to be used and stores them in a list.
- Broadcast: During epoch, app constantly broadcasts e_i. Other apps collect these broadcasted identifiers and persists them.
- Reporting: Upon positive diagnosis, user provides his / her app with the credential to upload to the server and the server publishes it.
- Status verification: Regularly, the app checks the newly uploaded identifier for the user and checks if it is an element of the list that is with the app. This is how the app determines whether the user is at risk or not.

Privacy Issues

1. Depseudonymization of Ephemeral Identifiers
2. Disclosing the social graph
3. Identifying diagnosed people
4. The opting choice

Privacy Issues

1. Depseudonymization of Ephemeral Identifiers

Centralized Systems

1. Major problem with centralized systems is the malicious usage of trapdoors. The ephemeral identifier can be mapped to the long term pseudonym of the user using the trapdoor.

2. Anyone who gets hold of the trapdoor can essentially collect identifiers unique to the pseudonym that's associated with the user.

3. Further, by exploiting the registration procedure, a corrupted authority can link the pseudonym to the actual user.

4. To counter, we can rotate keys when the trapdoor leaks. This can prevent major disasters for a limited period of time.

Privacy Issues

1. Depseudonymization of Ephemeral Identifiers Decentralized Systems

1. In decentralized system, there's no central authority to corrupt to in order to compromise the trapdoor. However, the ephemeral identifiers are derived from medium-term keys.

2. Anyone who gets access to these keys can link all identifiers. Report protocol uploads the medium-keys on the server.

3. The server is open access (due to decentralization) and therefore can tell whether two identifiers were from a key *k* or not.

4. Data mining can assist in reconstructing identities from ephemeral identifiers.

Privacy Issues

1. Depseudonymization of Ephemeral Identifiers

Vulnerabilities (Common Attacks)

1. In both centralized and decentralized systems, ephemeral identifiers are stored in the app for some time beyond the time they are broadcasted (called exposure time).

2. Exposure time in centralized systems is typically of the order of a day. In decentralized systems, ephemeral identifiers are typically of the order of two weeks.

3. These can be de-anonymized by *coercion (forceful access of phone contents)* and *theft (using malicious other apps or OSes to disclose contents of phone).*

Privacy Issues

1. Depseudonymization of Ephemeral Identifiers

Conclusion

1. Centralized systems concentrate to a single point of failure, i.e. the trapdoor. They put privacy under more risk than decentralized ones.

2. Decentralized systems have a clear advantage over centralized ones, when considering the risk of tracking people.

Privacy Issues
2. Disclosing the Social Graph
Centralized Systems

1. Exaggerated claim that centralized systems are bad in this case. However, there's no difference in decentralized & centralized as long as no report and status protocol is executed.

2. When all contacts are reported during report, leakage is for server only. If server is trusted & secure, we have no leakage!

3. A user who has never been in contact with any reported person will have no information revealed.

4. Through traffic analysis and comparison of reports, anonymity can be broken. Attack requires a lot of data mining (is infeasible).

Privacy Issues
2. Disclosing the Social Graph
Decentralized Systems

1. Attack works as: A reports identifiers he used when he met his colleague B but not the ones he used when he met his hated colleague, C.
   C with the help of B, can prove that A did not report everything. C can use this information to sue A.

2. System suffers from lack of *plausible deniability*.

3. This attack is harder to make at scale and could be detected.

Privacy Issues

2. Disclosing the Social Graph

Vulnerabilities (Common Attacks)

1. Coercion and theft can result in disclosure of social graph in both centralized and decentralized systems.

2. An adversary who has information about at-risk statuses (obtained by side channel attacks) for any two users, can deduce if A and B have both received an alert, that they have probably met some person in common.

3. This is highly risky in decentralized systems since they can publicly identify reported diagnosed users.

Privacy Issues

2. Disclosing the Social Graph

Conclusions

1. Centralized systems reveal to a determined malicious server a part of the social graph.

2. Decentralized systems reveal to a determined malicious user a proof of encounter with a diagnosed person.

Privacy Issues

3. Identifying Diagnosed People

Centralized Systems

1. If A (a user) registers a dummy account with which he only goes in contact with B, then using A's app with the report protocol will make A aware of whether B was reported or not.

2. Preventing this needs us to make sure that registration is not easily doable. This would prevent large scale attacks. Can be done by introducing CAPTCHAs to register accounts.

3. To tackle this ROBERT deactivates the account after reporting.

4. This shows a clear advantage of centralized systems over decentralized ones.

Privacy Issues
3. Identifying Diagnosed People
Decentralized Systems

1. Pseudonyms of reported diagnosed users are publicly revealed. Anyone can connect to the server and download a list of pseudonyms.

2. Therefore, anyone who has the ephemeral identifiers and information about whose identifiers they are, it could *a posteriori* recognize that this user was diagnosed and reported.

3. Paparazzi attack (extract identifier by boosting Bluetooth receiver), Nerd attack, Militia attack, and attack by organization (using hotels, shops, companies etc.'s Bluetooth collectors to collect information about visitors) are some attacks proposed against these systems.

4. Attacks can also be done by malicious apps that have several attack vectors like access to outgoing / incoming identifiers, access to other apps'. Operating systems or hardware can also be malicious. Some attacks coupled with video-surveillance techniques can also be used (mapping of identifier and pointer to recorded video).

Privacy Issues
3. Identifying Diagnosed People
Conclusions

1. Centralized systems have a clear advantage over decentralized ones in this case.

2. Decentralized systems could be protected by using PSI (private-set intersection), and several app protection schemes like not allowing apps running in the background access to Bluetooth broadcast.

3. Centralized systems could be with CAPTCHAs to prevent Sybil attacks.

Privacy Issues

4. The Opting Choice

Opting can have several forms:

a. Not installing the app at all: Faced with social pressure of not installing
b. Turning off Bluetooth
c. Using the app but refusing to report after diagnosis: Can result in coercion and irresponsible behavior perception by others. Twisted minded user can mimic the app but send random beacons and store nothing.

Privacy Issues

4. The Opting Choice

Mindset of a Grumpy User

1. A grumpy user is pressured to use the app (social / peer pressure).

2. In centralized systems, the grumpy user would receive an unsolicited alert from the server when someone who he came in contact with, has been diagnosed and reported. This would not cure his grumpiness since the server has identified him.

3. In decentralized systems, the grumpy user can let the app run at a small privacy cost. Whether the app raised an alert or not does not mean that someone reported him. This is better than centralized systems.

4. When a grumpy user is forced to report after a diagnosis, the grumpy user would fear that his identity and contacts are reported (centralized), or fear that his identity is revealed to an unknown adversary (decentralized)

Privacy Issues

4. The Opting Choice

Mindset of a Privacy-Freak Rational User

1. A rational user would only opt-in to the app because it brings a benefit.

2. A privacy-freak rational user might find the privacy cost of centralized systems larger than the cost of decentralized systems because of the risk to be deanonymized. Hence, the user would more likely opt-in to decentralized system solutions.

3. On the contrary, when it comes to reporting after diagnosis of this user, the privacy cost of decentralized systems is higher.

4. Centralized systems may generate more reports than a decentralized one. For decentralized systems to be more effective than centralized ones, we assume some form of pressure on users.

Privacy Issues

4. The Opting Choice

Mindset of a Malicious Privacy-Freak Rational User

1. This kind of user would be willing to modify the app in such a way that he gets benefit at a reduced cost of privacy.

2. In decentralized systems, the modified app would broadcast junk identifiers and report genuine ones. Thus, a (selfish) malicious privacy-freak user would generate no useful reports and no useful broadcasts.

3. In centralized systems, the user must broadcast (behavior like a honest user) his genuine identifiers but he may or may not perform reporting. Since the privacy cost to reporting is very small he could report and contribute.

Privacy Issues
4. The Opting Choice
Conclusion

1. Privacy-freak rational users would probably make decentralized contact tracing less effective than centralized contact tracing.

2. The lack of real free choice may result in more grumpy and decentralized-systems-inclined users.

Security Issues

1. False Encounter: The Lazy Student Attack
2. False Report: The Terrorist Attack

Security Issues

1. False Encounter: The Lazy Student Attack

1. Both systems have equal potential danger of replay and relay attacks. Adversary A that has caught identifiers from an app (say A), can replay them to another app (say C). C may store these identifiers even though it hasn't encountered A.

2. By hunting for identifiers (best place would be a hospital). Attack works for both centralized and decentralized systems, but it can scale in decentralized systems with dark economy.

3. We can use a combination of location identifier, random number and hashing to prevent replay attacks. However, it creates a problem of undeniable evidence of geographic positions (especially in decentralized systems where this is posted on a blockchain). Raises privacy concerns.

Security Issues

2. False Report: The Terrorist Attack

1. Malicious and diagnosed user could upload forged identifiers to the server in the report protocol. There is no way for a server to know if an app is honestly reporting or not.

2. In a grouped attack, a group of people are using a modified app to pool all the information. If a person from this group is diagnosed, app would report all information to alert all members of this group, to create a mass alert.

3. In a trolling attack (simpler form of terrorist attack), someone diagnosed or about to be diagnosed makes his phone circulate in a crowd. This creates fake encounters. After the person is diagnosed, he can report and trigger alerts for many people.

A Third Way

- Centralized systems suffer from the use of trapdoor, while decentralized systems suffer from that status requires the central server universally readable.

- Strengthening strategies like PSI, re-randomization of tokens (server cannot link them by observations / data mining), private messaging could be used.

- Some research directions - Centralized architecture with open-access server, decentralized architecture with restricted-access server, decentralized architecture reporting received identifiers and public-key based architecture.

A Third Way

Centralized Architecture with Open-Access Server

1. We use an ephemeral identifier pair (public-secret). These pairs are anonymously uploaded to the server beforehand.

2. Public identifiers are broadcasted. Secret identifier is persisted in the user's device. On reporting, the server publishes the secret identifier and each user can check if their secret identifier matches to the published one.

3. This uses a hybrid architecture. It builds up an open-access server on top of a centralized one

A Third Way

Decentralized Architecture with Restricted-Access Server

1. This approach builds an access-restricted server on top of a centralized one. The server is no longer universally readable.

2. It makes the status protocol private. It is a 2-party protocol with two sets of input (S = set of the server and I = set of identities), that returns to the app the at-risk status of the user.

3. By running the status protocol multiple times and getting $I \cap U$, an adversary can leak diagnosed people. We can prevent this by limiting # of queries made.

4. A perfectly private approach should ensure that neither the contents of the intersection of S and I, nor the cardinality of this set is revealed. This can be achieved by using *Bloom filters (linear complexity) or Flaojet-Martin sketch (logarithmic complexity).*

A Third Way

Decentralized Architecture Reporting Received Identifiers

1. The authors of PACT-West proposed to report the list of received identifiers instead of the list of sent ones.

2. We use public-secret identifier pairs here. This technique uses re-randomization of public ephemeral identifiers that can be matched to their secret ones.

3. This approach does not solve the Sybil attack issue since malicious user can create dummy secret identifiers to identify diagnosed people.

A Third Way

Public-Key-Based Architecture

1. Each user anonymously posts on a blockchain their ephemeral keys and memorizes the address of the block on which they appear.

2. This allows Bluetooth beacons to compactly share D-H keys by simply broadcasting the address.

3. Every user can establish a shared D-H key with the posted public keys and his / her own secret key.

4. Again, the Sybil attack can be performed by creating dummy D-H pairs. This can be limited to some extent by limiting the # of D-H keys a user can register.

Conclusion

1. From all the privacy and security risks that we saw, neither decentralized nor centralized solutions are better than the other. Both of them have their merits and demerits.

2. Centralized systems face the risk of revealing identities of users and their reported contacts, but the adversary can only be the server. Decentralized systems face the risk of revealing identities of diagnosed people, but the adversary can be anyone.

3. We need the best of the two worlds. Therefore, the hybrid directions that exist look promising.

Questions

1. Do you think the research in the privacy and security of contact tracing should continue even after the COVID-19 pandemic is over?

2. Do you think a universal standard for implementing contact tracing protocols should be adopted? If yes, what challenges would we face at an international level?

3. Just like mask mandates and vaccine mandates were proposed, do you think contact tracing app *opt-in* mandates will be introduced? What are the ethical challenges in this?

4. This paper didn't talk about any cryptographic flaws in the underlying algorithms that were used in both systems. Do you think the design of cryptographic contact tracing algorithms would behave differently based on the system (centralized / decentralized) in which they are implemented?