Bulletin of the Technical Committee on

Data Engineering

June 2020 Vol. 43 No. 2

IEEE Computer Society

Letters

Opinions

Special Issue on Data Technologies Behind Digital Contact Tracing for COVID19

Conference and Journal Notices

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TCDE web page is http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

One of the beauties of the Data Engineering Bulletin, with a history of 43 years and 157 issues, is that it chronicles how topics of database research evolve and sometimes reinvent themselves over time. Phil Bernstein's opinion piece in this issue, titled "Resurrecting Middle-Tier Distributed Transactions," is another testimony to this beauty. Bernstein tells an interesting story of transaction processing monitors running on middle-tier servers, and predicts the return of middle-tier distributed transactions to the mainstream after a 15-year decline.

Guoliang Li put together the current issue consisting of 6 papers on the interactions between database systems and AI. This is a fascinating topic. Traditional databases are heavily optimized monolithic systems designed with heuristics and assumptions. But recent work has shown that critical data structures such as database indices are merely models, and can be replaced with more flexible, faster, and smaller machine learned models such as neural networks. This opens the door to using data driven approaches for system design. On the other hand, deep learning is still facing the challenge in incorporating database accesses in end-to-end training, which hampers the use of existing structured knowledge in learning.

Haixun Wang
WeWork Corporation

Letter from the Special Issue Editor

Missing!

Joseph Gonzalez
University of California at Berkeley

Resurrecting Middle-Tier Distributed Transactions

Philip A. Bernstein
Microsoft Research, Redmond, WA 98052

1 Introduction

Over the years, platforms and application requirements change. As they do, technologies come, go, and return again as the preferred solution to certain system problems. In each of its incarnations, the technology's details change but the principles remain the same. One such technology is distributed transactions on middle-tier servers. Here, we argue that after a 15-year decline, they need to return to the mainstream.

In the 1980's, Transaction Processing (TP) monitors were a popular category of middleware product that enabled customers to build scalable distributed systems to run transactions. Example products were CICS (IBM), Tuxedo (AT&T for Unix), ACMS (DEC for VAX/VMS), and Pathway (Tandem for Guardian) [4]. Their main features were multithreaded processes (not supported natively by most operating systems), inter-process communication (usually a crude form of remote procedure call), and a forms manager (for end users to submit transaction requests). The TP monitor ran on middle-tier servers that received transaction requests from front-end processors that communicated with end-user devices, such as terminals and PC's, and with back end database servers. The top-level application code executed on the middle-tier and invoked stored procedures on the database server.

In those days, database management systems (DBMS's) supported ACID transactions, but hardly any of them supported distributed transactions. The TP monitor vendors saw this as a business opportunity and worked on adding a transaction manager feature that implemented the two-phase commit protocol (2PC). Such a feature required DBMS's to expose Start, Prepare, Commit, and Abort as operations that could be invoked by the TP monitor. Unfortunately, most of them didn't support Prepare, and even if they did, they didn't expose it to applications. They were willing to do so, but they didn't want to implement a different protocol for each TP monitor product. Thus, the XA standard was born, which defined TP monitor and DBMS interfaces (including Prepare) and protocols that allowed a TP monitor to run a distributed transaction across DBMS servers [17].

This middle-tier architecture for distributed transactions was popular for about 20 years, into the late 1990s. Then TP monitors were replaced by Application Servers, which integrated a TP monitor with web servers, so it could receive transaction requests over HTTP, rather than receiving them from devices connected by a local area or terminal network. Examples include Microsoft Transaction Server, later renamed COM+, and Java Enterprise Edition (JEE), implemented by IBM's WebSphere Application Server, Oracle's WebLogic Application Server, and Red Hat's JBoss Application Server [12]. The back end architecture was the same as before. Each transaction started executing on a middle-tier server and invoked stored procedures to read and write the database.

Although this execution model is still widely used, starting in the early 2000's it fell out of favor for new application development, especially for applications targeted for cloud computing. More database vendors offered built-in support for distributed transactions, so there was less need to control the distributed transaction from the middle tier. A larger part of database applications executed on data that was cached in the middle tier. And the NoSQL movement argued that distributed transactions were too slow, that they limited scalability, and that customers rarely needed them anyway [11]. Eventual consistency became all the rage [18].

The critics of distributed transactions had some good points. But in the end, developers found that mainstream

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

programmers really did need ACID transactions to make their applications reliable in the face of concurrent access to shared data and despite server failures. Thus, some NoSQL (key-value) stores added transaction support (e.g., CosmosDB [2], DynamoDB [8]). Google, which had initially avoided support for multi-row distributed transactions in Bigtable [5], later introduced them in Spanner [6]. There are now many cloud storage services and database products that support distributed transactions.

Like product developers, database researchers have also focused on distributed transactions for back-end database systems. Almost universally, they assume that transactions execute as stored procedures and that middle-tier applications invoke those stored procedures but do not execute the transaction logic themselves.

2 Stateful Middle-Tier Applications

This focus on stored procedures is well justified by the needs of traditional TP applications. However, stored procedures are not a good way of encapsulating application logic for a growing fraction of stateful applications that run on the middle tier. These include multi-player games, datacenter telemetry, Internet of Things, and social and mobile applications. Objects are a natural model for the entities in these applications, such as games, players, datacenter hardware, sensors, cameras, customers, and smart phones. Such applications have a large number of long-lived stateful objects that are spread over many servers and communicate via message passing. Like most new applications, these applications are usually developed to run on cloud computing platforms.

These applications typically execute on middle-tier servers, rather than as stored procedures in database servers. They do this for many reasons. They need large main memory for the state of long-lived objects. They often have heavy computation needs, such as rendering images or computing over large graphs. They use a lot of computation for message passing between objects so they can scale out. And they need computation to be elastic, independent of storage requirements. These needs are satisfied by compute servers that are cheaper than database servers because they have less storage. Hence, these apps run on compute servers in the middle tier.

2.1 Requirements for Mid-Tier Cloud Transactions

Some middle-tier applications need transactions because they have functions that read and write the state of two or more stateful objects. For example, a game may allow users to buy and sell virtual game objects, such as weapons, shields, and vehicles. A telemetry application may need to process an event exactly once by removing it from a queue and updating telemetry objects based on that event. A social application may need to add a user to a group and modify the user's state to indicate membership in that group. Each of these cases needs an ACID transaction over two or more objects, which may be distributed on different servers. Since these applications are usually developed to run on cloud computing platforms, distributed transaction support must be built into the cloud platform, a capability that is rarely supported today for cloud computing.

Distributed transactions for middle-tier applications on a cloud computing platform have four requirements that differ from those supported by the late-1990's products that run transactions on the middle-tier. First, like all previous transaction mechanisms, they need to offer excellent performance. But unlike previous mechanisms, it's essential that they be able to scale out to a large number of servers, leading to the first requirement: The system must have high throughput and low transaction latency, at least when transactions have low contention, and in addition must scale out to many servers.

To scale computation independently of storage, these applications typically save their state in cloud storage. The developers' choice of cloud storage service depends on their application's requirements (e.g., records, documents, blobs, SQL), their platform provider's offerings (e.g., AWS, Azure, Google), their employer's storage standards, and their developers' expertise. Thus, we have this second requirement: The transaction mechanism must support applications that use any cloud storage service.

The transaction mechanism needs persistent storage to track transaction state: started, prepared, committed,

or aborted. Like the apps themselves, it needs to use cloud storage for this purpose, which is the third requirement: The transaction mechanism must be able to use any of the cloud storage services used by applications.

The traditional data structure for storing transaction state is a log. The transaction manager relies on the order of records in the log to understand the order in which transactions executed. Although cloud vendors implement logs to support their database services, they do not expose database-style logging as a service for customers, leading to a fourth requirement: The transaction mechanism cannot rely on a shared log, unless it implements the log itself, in which case the log must run on a wide variety of storage services.

Due to the latency of cloud storage, requirements (2)-(4) create challenges in satisfying requirement (1).

The above requirements are a first cut, based on today's applications and platforms. It is also worth targeting variations. For example, requirement (1) could include cost/performance, which might require a tradeoff against scalability. And (4) might go away entirely if cloud platforms offer high-performance logging as a service.

3 An Implementation in the Orleans Framework

The rest of this paper sketches a distributed transaction mechanism that satisfies the above requirements [9]. Our group built it for Microsoft's actor-oriented programming framework, called Orleans, which is open source and runs on both Windows and Linux [16]. The distributed transaction project is part of a longer-term effort to enrich Orleans with other database features to evolve it into an actor-oriented database system that supports geo-distribution, stream processing, indexing, and other database features [3].

3.1 Two-Phase Commit and Locking

For ACID semantics, Orleans transactions use two-phase commit (2PC) and two-phase locking (2PL). Our first challenge was to obtain high throughput and scalability despite the requirement to use cloud storage. In our runs, a write to cloud storage within a datacenter takes 20 ms and has high variance. With 2PC, a transaction does two synchronous writes to storage. Therefore, if 2PL is used, a transaction holds locks for 40ms, which limits throughput to 25 transactions/second (TPS). Low-latency SSD-based cloud storage is faster, but still incurs over 10 ms latency, plus higher cost. To avoid this problem, we extended early lock release to 2PC [1, 7, 10, 13, 14, 15]. After a transaction T1 terminates, it releases locks before writing to storage in phase one of 2PC. This allows a later transaction T2 to read/update T1's updated objects. Thus, while T1 is writing to storage, a sequence of later transactions can update an object, terminate, and then unlock the object. To avoid inconsistency, the system delays committing transactions that directly or indirectly read or overwrite T1's writeset until after T1 commits. And if T1 aborts, then those later dependent transactions abort too. Using this mechanism, we have seen transaction throughput up to 20x that of strict 2PL/2PC.

3.2 Logging

Our initial implementation used a centralized transaction manager (TM) per server cluster [9]. It ran on an independent server and was multithreaded. Since message-passing is a potential bottleneck, it batched its messages to transaction servers. It worked well with throughput up to 100K TPS. However, it had three disadvantages: it was an obvious bottleneck for higher transaction rates; a minimum configuration required two servers (i.e., primary and backup TM) in addition to servers that execute the application; and it added configuration complexity since TM servers did not run Orleans and thus had to be deployed separately from application servers.

These disadvantages led us to redesign the system to avoid a centralized TM. Instead, we embed a TM in each application object. Each TM's log is piggybacked on its object's storage. This TM-per-object design avoids the above disadvantages and improves transaction latency by avoiding roundtrips to a centralized TM. However, it doesn't work for objects that have no updatable storage. For example, an object that performs a money transfer calls two stateful objects, the source and target of the transfer, but it has no state itself. We allow such an object to

participate in a transaction by delegating its TM function to a stateful participant in the transaction, that is, one that has updatable storage.

Orleans transactions write object state to a log to enable undo when a transaction aborts. This is impractical for large objects and is a poor fit for concurrency control that exploits operation commutativity. We therefore developed a prototype that logs operations.

4 Summary

Many new cloud applications run their logic on the middle tier, not as stored procedures. They need distributed transactions. Thus, cloud computing platforms can and should offer scalable distributed transactions.

5 Acknowledgments

I'm grateful to the team that built the initial and final implementations of Orleans transactions: Jason Bragg, Sebastian Burckhardt, Tamer Eldeeb, Reuben Bond, Sergey Bykov, Christopher Meiklejohn, Alejandro Tomsic, and Xiao Zeng. I also thank Bailu Ding and Dave Lomet for suggesting many improvements to this paper.

References

[1] Athanassoulis, Manos ; Johnson, Ryan ; Ailamaki, Anastasia ; Stoica, Radu, Improving OLTP Concurrency through Early Lock Release, EPFL-REPORT-152158, https://infoscience.epfl.ch/record/152158?ln=en, 2009.

[2] Azure CosmosDB, https://azure.microsoft.com/en-us/services/cosmos-db/

[3] Bernstein, P.A., M., T. Kiefer, D. Maier: Indexing in an Actor-Oriented Database. CIDR 2017

[4] Bernstein, P. A., E. Newcomer: Chapter 10: Transactional Middleware Products and Standards, in Principles of Transaction Processing, Morgan Kaufmann, 2nd ed., 2009.

[5] Chang, F., J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, R.E. Gruber: Bigtable: A Distributed Storage System for Structured Data. ACM Trans. Comput. Syst. 26(2): 4:1-4:26 (2008)

[6] Corbett, J.C. et al: Spanner: Google's Globally Distributed Database. ACM Trans. Comput. Syst. 31(3): 8:1-8:22 (2013)

[7] DeWitt, D.J., R.H. Katz, F. Olken, L.D. Shapiro, M. Stonebraker, D.A. Wood: Implementation Techniques for Main Memory Database Systems. SIGMOD 1984: 1-8

[8] DynamoDB, https://aws.amazon.com/dynamodb/

[9] Eldeeb, T. and P. Bernstein: Transactions for Distributed Actors in the Cloud. Microsoft Research Tech Report MSR-TR-2016-1001.

[10] Graefe, G., M. Lillibridge, H. A. Kuno, J. Tucek, A.C. Veitch: Controlled lock violation. SIGMOD 2013: 85-96

[11] Helland, P., Life beyond Distributed Transactions: an Apostate's Opinion. CIDR 2007: 132-141

[12] Java EE documentation, http://www.oracle.com/technetwork/?java/javaee/documentation/index.html

[13] Larson, P-A, et al.: High-Perf. Concurrency Control Mechanisms for Main-Memory Databases. PVLDB 2011

[14] Levandoski, L.J., D.B. Lomet, S. Sengupta, R. Stutsman, R. Wang: High Performance Transactions in Deuteronomy. CIDR 2015

[15] David B. Lomet: Using Timestamping to Optimize Two Phase Commit. PDIS 1993: 48-55

[16] Orleans, http://dotnet.github.io/orleans

[17] The Open Group, Distributed Transaction Processing: The XA Specification, http://pubs.opengroup.org/onlinepubs/009680699/toc.pdf.

[18] Vogels W., Eventually Consistent. ACM Queue 6(6): 14-19 (2008)

english]babel utf8]inputenc colorlinks]hyperref

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

PACT: Privacy-Sensitive Protocols And Mechanisms for Mobile Contact Tracing

Justin Chan[1], Dean Foster[5], Shyam Gollakota[1], Eric Horvitz[†,1,3,4], Joseph Jaeger[†,1], Sham Kakade[†,1,2], Tadayoshi Kohno[1], John Langford[†,4], Jonathan Larson[4], Puneet Sharma[6], Sudheesh Singanamalla[1], Jacob Sunshine[3], Stefano Tessaro[†,1]

[1] Paul G. Allen School of Computer Science & Engineering, University of Washington
[2] Department of Statistics, University of Washington
[3] School of Medicine, University of Washington
[4] Microsoft Research
[5] University of Pennsylvania
[6] Boston Public Health Commission
[†] Corresponding authors*

Abstract

The global health threat from COVID-19 has been controlled in a number of instances by large-scale testing and contact tracing efforts. We created this document to suggest three functionalities on how we might best harness computing technologies to supporting the goals of public health organizations in minimizing morbidity and mortality associated with the spread of COVID-19, while protecting the civil liberties of individuals. In particular, this work advocates for a third-party–free approach to assisted mobile contact tracing, because such an approach mitigates the security and privacy risks of requiring a trusted third party. We also explicitly consider the inferential risks involved in any contract tracing system, where any alert to a user could itself give rise to de-anonymizing information.

More generally, we hope to participate in bringing together colleagues in industry, academia, and civil society to discuss and converge on ideas around a critical issue rising with attempts to mitigate the COVID-19 pandemic.

1 Introduction and Motivation

Several communities and nations seeking to minimize death tolls from COVID-19, are resorting to *mobile-based, contact tracing technologies* as a key tool in mitigating the pandemic. Harnessing mobile computing technologies is an obvious means to dramatically scale-up conventional epidemic response strategies to do tracking at population scale. However, straightforward and well-intentioned contact-tracing applications can invade personal privacy and provide governments with justification for data collection and mass surveillance that are inconsistent with the civil liberties that citizens will and should expect—and demand. To be effective, acceptable, and consistent with the need to observe commitments to privacy, we must leverage designs and computing advances in privacy and security. In cases where it is valuable for individuals to share data with

*Email: jsjaeger@cs.washington.edu, eric@horvitz.org, sham@cs.washington.edu, jl@hunch.net, tessaro@cs.washington.edu

others, systems must provide voluntary mechanisms in accordance with ethical principles of personal decision making, including disclosure, and consent. We refer to efforts to identify, study, and field such privacy-sensitive technologies, architectures, and protocols in support of mobile tracing as PACT (*P*rivacy sensitive protocols *A*nd mechanisms for mobile *C*ontact *T*racing).

The objective of PACT is to set forth transparent privacy and anonymity standards, which permit adoption of mobile contract tracing efforts while upholding civil liberties.

This work specifies a third-party–free set of protocols and mechanisms in order to achieve these objectives. While approaches which rely on trusted third parties can be straightforward, many naturally oppose the aggregation of information and power that it represents, the potential for misuse by a central authority, and the precedent that such an approach would set.

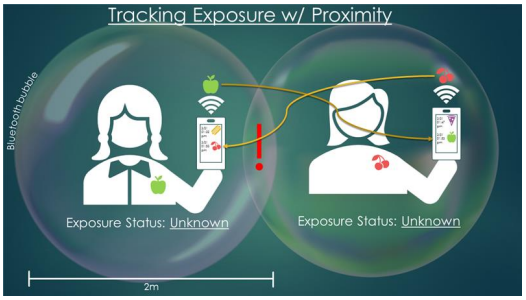


Figure 1: Proximity-Based Tracing. The basic idea is that users broadcast signals ("pseudonyms"), while also recording the signals they receive. Notably, this *co-location* approach avoids the need to collect and share *absolute* location information. Credit: M Eifler.

It is first helpful to review the conventional contact tracing strategies executed by public health organizations, which operate as follows: Positively tested citizens are asked to reveal (voluntarily, or enforced via public health policy or by law depending on region) their contact history to public health officers. The public health officers then inform other citizens who have been at risk to the infectious agent based on co-location, via some definition of co-location, supported by look-up or inference about locations. The citizens deemed to be at risk are then asked to take appropriate action (often to either seek tests or to quarantine themselves and to be vigilant about symptoms). It is important to emphasize that the current approach *already* makes a tradeoff between the privacy of a positively tested individual and the benefits to society.

We describe mobile contact-tracing functionalities that seeks to augment the services provided by public health officers, by enabling the following capabilities via computing and communications technology:

- **Mobile-assisted contact tracing interviews:** A citizen who becomes ill can use this functionality to improve the efficiency and completeness of manual contact tracing interviews. In many situations, the citizen can speed up the interview process by filling in much of a contact interview form before the contact interview process even starts, reducing the burden on public health authorities. The privacy-sensitivity here is ensured since all the data remains on the user's device, except for what they voluntarily decide to reveal to health authorities in order to enable contact tracing. In advance of their making a decision to share, they are informed about how their data may be used and the potential risks of sharing.

- **Narrowcast messages:** Public health authorities can make available custom-tailored messages to specific, relevant subsets of citizens. For example, the following message might be issued: "If you visited the X Eldercare Center between March 7th and 10th, please email yy@hhhealth.org" or "Please refrain from entering playground Z until April 6th because it needs to undergo decontamination." A mobile app can download all of these messages and display those relevant to a citizen based on the app's sensory log or potential future movements. This capability allows public health officials to quickly warn people when new hotspots arise, or canvas for general information. It enables a citizen to be well-informed about extremely local pandemic-relevant events.

- **Privacy-sensitive, mobile tracing:** Proximity-based signals seem to provide the best available contact sensor from one phone to another; see Figure 1 for the basic approach. Proximity-based sensing can

be done in a privacy-sensitive manner. With the approach, no absolute location information is collected nor shared. Variants of proximity-based analyses have been employed in the past for privacy-sensitive analyses in healthcare [23]. Taking advantage of proximity-based signals can speed the process of contact discovery and enable contact tracing of otherwise undiscoverable people like the fellow commuter on the train. This can also be done with a third-party–free approach providing similar privacy tradeoffs as manual contact tracing. This functionality can enable someone who has become ill with symptoms consistent with COVID-19, or who has received confirmation of infection with a positive test for COVID-19, to voluntarily and under a pseudonym, share information that may be relevant to the wellness of others. In particular, a system can manage, in a privacy-sensitive manner, data about individuals who came in close proximity to them over a period of time (e.g., the last two weeks), even if there is no personal connection between these individuals. Individuals who share information do so with disclosure and consent around potential risks of private information being shared. We further discuss disclosure, security concerns, and re-identification risks in Section 2.

Importantly, these protocols, by default, keep *all* personal data on a citizens' phones (aside for pseudonymous identifiers broadcast to other local devices), while enabling these key capabilities; information is shared via voluntary disclosure actions taken, with the understandings relayed via careful disclosure. For example, if someone never tests positive for COVID-19 or tests positive but decides not to use the system, then *NO* data is ever sent from their phone to any remote servers; such individuals would be contacted by standard contact tracing mechanisms arising from reportable disease rules. The data on the phone can be encrypted and can be set up to automatically time out based on end-user controlled policies. This would prevent the dataset from being accessed or requested via legal subpoena or other governmental programs and policies.

We specify protocols for all three separate functionalities above, and each app designer can decide which ones to use. These protocols notably have different value adoption curves: Narrowcast and Mobile-assisted contact tracing have a value which is linear in the average adoption rate while privacy-sensitive mobile tracing has value quadratic in the average adoption rate due to requiring both ends of the connection be working. This quadratic dependence implies low initial value so we expect Narrowcast and Mobile-assisted contact tracing to provide initial value in adoption while privacy-sensitive mobile tracing provides substantial additional value once adoption rates are high.

We note that there are an increasing number of concurrent contact tracing protocols being developed – see in particular Section 5 for a discussion of solutions based on proximity based tracing (as in Figure 1). In particular, there are multiple concurrent approaches using proximity based signaling; our approach has certain advantageous properties, as it is particularly simple and requires very little data transfer.

One point to emphasize is that, with this large number of emerging solutions, it is often difficult for the user to interpret what "privacy preserving" means in many of these protocols[1]. One additional goal in providing the concrete protocols herein is to have a broader discussion of both privacy-sensitivity and security, along with a transparent discussion of the associated re-identification risks — the act itself of alerting a user to being at risk provides de-anonymizing information, as we discuss shortly.

From a civil liberties standpoint, the privacy guarantees these protocols ensure are designed to be consistent with the disclosures already extant in contract tracing methods done by public health services (where some information from a positive tested citizen is revealed to other at risk citizens). In short, we seek to empower public health services, while maintaining civil liberties.

We also note that these contact tracing solutions are not meant to replace conventional contact tracing strategies employed by public health organizations; not everyone has phones, and not everyone that has a phone will use this app. Therefore, it is still critical to leverage conventional approaches, along with the approaches

[1] In fact, due to re-identification risks, there is a strong case to be made that the terminology of "privacy preserving" is ill-suited to this context.

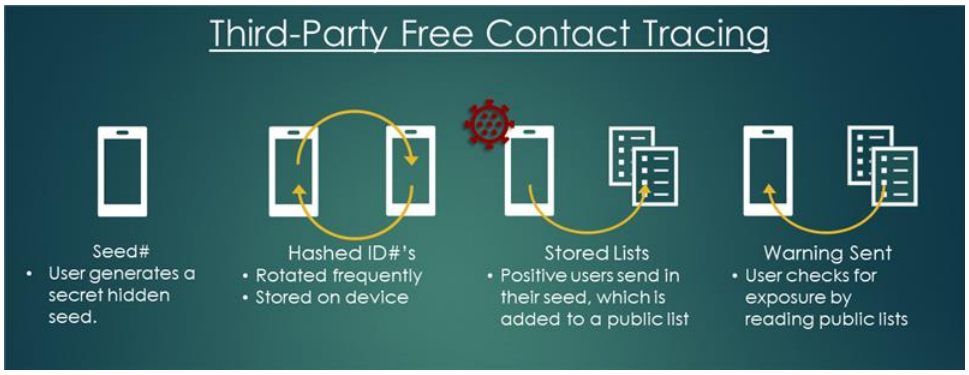


Figure 2: PACT Tracing Protocol. First, a user generates a random seed, which they treat as private information. Then all users broadcast random-looking signals to users in their proximity via Bluetooth and, concurrently, all users also record all the signals they hear being broadcast by other users in their proximity. Each person's broadcasts (their "pseudonyms") are a function of their private seed, and they change these broadcasted pseudonyms periodically (e.g. every minute). Whenever a user tests positive, the positive user then can voluntarily publish, on a public server, information which enables the reconstruction of all the signals they have broadcasted to others during the infection window (precisely, they publish their private seed, and, using the seed, any other user can figure out what pseudonyms the positive user has previously broadcasted). Now, any other user can determine whether they are at risk by checking whether the signals they heard are published on the server. Note that the "public lists" can be either lists from hospitals, which have confirmed seeds from positive users, or they can be self-reports (see Section 2.3). Credit: M Eifler.

outlined in this paper. In fact, two of our protocols are designed for assisting public health organizations (and are designed with input from public health organizations).

2 FAQ: Privacy, Security, and Re-Identification

Throughout, we refer to an *at risk* individual as one who has been in contact with an individual who has tested as positive for COVID-19 (under criteria as defined by public health programs, e.g., "within 6 feet for over 10 minutes").

Before we start this discussion, it is helpful to consider one principle which the proposed protocols respect: "If you do not report as being positive, then no information of yours will leave your phone." From a more technical standpoint, the statement that is consistent with our protocols is:

If you do not report as being positive, then only random ("pseudonymized") signals are permitted to be broadcast from your phone.

These random broadcasts are what allows proximity based tracing; see Figure 2 for a description of the mobile tracing protocol. It is worthwhile to note that this principle is consistent, in spirit, with conventional contract tracing approaches, where only positively tested individuals reveal information to the public health authorities. With the above principle, the discussion at hand largely focuses on what can be inferred when a positive disclosure occurs along with how a malicious party can impact the system.

We focus the discussion on the "mobile tracing" protocol for the following reasons: "Narrowcasting" allows people to listen for events in their region, so it can viewed as a one way messaging system. For "mobile-assisted interviews," all the data remains on the user's device, except for what they voluntarily reveal to public health authorities in order to enable contact tracing.

All the claims are consequences of basic security properties that can formally be proved about the protocol, and in particular, about the cryptographic mechanism generating these random-looking signals.

2.1 Confidentiality, Re-Identification, and Inferential Risks

We start first with what private information is protected and what is shared voluntarily, following disclosure and consent. The inferential risk is due to that the alert itself is correlated with other information, from which a user could deduce de-anonymizing information.

1. If I tested positive and I voluntarily disclose this information, what does the protocol reveal to others?

   Any other citizen who uses a mobile application following this protocol who has been at risk is notified. In some versions the time(s) that the exposure(s) occurred may be shared. In the basic mobile tracing system that we envision, beyond exposure to specific individuals, no information is revealed to any other citizens or entities (authorities, insurance companies, etc).

   It is also worthwhile noting that, if you are negative, then the protocol does not directly transmit *any* of your private information to any public database or any other third party; the protocol does transmit random ("pseudonymized") signals that your phone broadcasts.

2. **Re-Identification and Inferential Risks.** Can a positive citizen's identity, who chooses to report being positive, be *inferred* by others?

   Identification is possible and is a risk to volunteers who would prefer to remain de-identified. Preventing proximity-based identification of this sort is not possible to avoid in any protocol, even in manual contact tracing as done by public health services, simply because the exposure alert may contain information that is correlated with identifying information. For example, an individual who had been in close proximity to only one person over the last two weeks can infer the identity of this positively tested individual. However, the positive's identity will never be explicitly broadcast. In fact, identities are not even stored in the dataset: It is only the positive person's random broadcasts that are stored.

3. **Mitigating Re-identification.** Can the app be designed so as to mitigate re-identification risks to average users?

   While the protocol itself allows a sophisticated user, who is at risk, to learn the time at which the exposure occurred, the app itself can be designed to mitigate the risk. For example, in the app design, the re-identification risk could be mitigated by only informing the user that they are at risk, or the app could only provide the rough time of day at which the exposure occurred. This is a mild form of mitigation, which a malicious or sophisticated user could try to circumvent.

2.2 Attacks

We now directly address questions about the potential for malicious hackers, governments, or organizations to compromise the system. In some cases, cryptographically secure procedures can prevent certain attacks, and, in other cases, malicious disclosure of information is prevented because the protocol stores no data outside of your device by default. Only cryptographically secure data from positively confirmed individuals is stored outside of devices.

1. **Integrity Attacks.** If you are negative, can a malicious citizen listen to your phone's broadcasts and, then report positive pretending to be you?

   No, this is not possible, provided you keep your initial seed private (see Figure 2). Furthermore, even if the malicious party records all Bluetooth signals going into and out of your phone, this is not possible.

This attack is important to avoid, since, suppose a malicious entity observes all Bluetooth signals sent from your phone. Then, you would not want this entity to report you as positive. This attack is not possible as the seed uniquely identifies your broadcasts and remains unknown to the attacker, unless the attacker is able to successfully break the underlying cryptographic mechanism, which is unlikely to be possible.

2. **Inferential Attacks.** Can a positive citizen's location, who chooses to report being positive, be inferred by others?

   It is possible for a malicious party to simultaneously record broadcasts at multiple different locations, including those that the positive citizen visited. Using these recordings, the malicious party could infer where the positive citizen was. The times at which the citizen visited these locations can also be inferred.

3. **Replay and Reliability Attacks.** If a citizen is alerted to be at risk, is it possible the citizen was not in the proximity of a positive individual?

   There are a few unlikely attacks that can trigger a false alert. One is a *replay attack*. For example, suppose a malicious group of multiple individuals colludes to try and pretend to be a single individual; precisely, suppose they all use the same private seed (see Figure 2). Then if only one of these malicious individuals makes a positive report, then multiple people can be alerted, even if those people were not in the proximity of the person who made the positive report. The protocol incorporates several measures to make such attacks as difficult as possible.

4. **Physical Attacks.** What information is leaked if a citizen's device is compromised by a hacker, stolen, or physically seized by an authority?

   Generally, existing mechanisms protect access to the storage of a phone. Should these mechanisms fail, the device only stores enough information to reconstruct the signals broadcast over a period of time prior to the compromise which amounts to the length of the infection window (i.e., two weeks), in addition to collected signals. This enables some additional inference attacks. It is not possible to learn whether the user has ever reported positive.

2.3 Lab-Based and Self-Confirmed Reporting; Reliability Concerns

Given that we would like the protocol to be of use to different states and countries, we seek an approach which allows for both security in reporting and for flexibility from the app designer in regions where it may make sense to consider reports which are self-confirmed positives tests or self-confirmed symptoms.

1. **Reporting.** Does the protocol support both medically confirmed positive tests and self-confirmed positives tests?

   Yes, it supports both. The uploaded files contain signatures from the uploading party (i.e. from a hospital lab or from any app following the protocol). This permits an app designer the freedom to use information from health systems and information from individuals in possibly different manners. In less developed nations, it may be helpful to permit the app designer to allow for reports based on less reliable signatures.

2. **Reliability.** How will the protocol handle issues of false positives and false negatives, with regards to alerting? What about cases when users don't have (or use their) mobile phones?

   The protocol does not explicitly address this, but a deployment requires both thoughtful app design and responsbile communication with the public.

   With regards to the former, the false positive and false negative rates have to be taken into account when determining how to make at risk reports. More generally, estimates of the probabilities can be helpful to a user

(or an otherwise interpretable report); such reports can be particularly relevant for those in high risk categories (such as the elderly and immuno-compromised individuals).

Furthermore, not everyone has a smartphone, and not everyone with a smartphone will use this app. Thus, users of this app – if they have not received any notification of exposure with COVID-19 positive cases – should not assume that they have not been around such positive cases. This means, for example, that they should still be cautious and follow all appropriate current public health guidelines, even if the app has not alerted them to possible COVID-19 exposure. This is particularly important until there is sufficient penetration of the app in any local population.

2.4 Other Threats (Exogenous to the Protocols)

We now list threats that are outside of the scope of the protocol, yet important to consider. Care should be taken to address these concerns:

- **Trusted Communication.** Communication between users and servers must be protected using standard mechanisms (i.e., the TLS protocol [18]).
- **Spurious Entries.** Self-reporting allows a malicious user to report themselves positive when they are not, and generally may allow several fake reports (i.e. a flooding attack). Mitigation techniques should be introduced to reduce the risk of such attacks.
- **Invalid Authentication.** Positive reports should be validated using digital signatures, e.g., by healthcare providers. This requires appropriate public-key infrastructure to be in place. Additional vulnerabilities related to misuse or misconfiguration of this infrastructure can affect reliability of positive reports.
- **Implementation Issues.** Implementation aspects may weaken some of our claims, and need to be addressed. For example, signals we send over Bluetooth as part of our protocol may be correlated with other signals which de-anonymize the user.

3 Protocols

We now provide an overview of the three functionalities of PACT.

3.1 Privacy-sensitive Mobile Tracing

This section describes and discusses a privacy-sensitive mobile tracing protocol. Our protocol follows a pattern wherein users exchange IDs via Bluetooth communication. If a user is both infected (we refer to such users as *positive*, and otherwise as *negative*) and willing to warn others who may have been at risk via proximity to the user, then de-identified information is uploaded to a server to warn other users of potential exposure. The approach has been followed by a number of similar protocols – we describe the differences with some of them in Section 5. In Appendix B, we discuss an alternative approach which may offer some efficiency and privacy advantages, at the cost of relying on signatures as opposed to hash functions.

3.1.1 Protocol description

Low-level technical details are omitted, e.g., how values are broadcast. Further, it is assumed the communication between users and the server is protected using the Transport Layer Security (TLS) protocol. We first describe a variant of the protocol *without* entry validation, and discuss how to easily extend it to validate entries below.

- **Parameters.** We fix an understood time unit dt and define Δ such that $\Delta \cdot \mathrm{dt}$ equals the infection window. (Typically, this would be two weeks.) We also fix the bit length n of the identifiers. (Typically, $n = 128$.) We also use a function $G : \{0,1\}^n \to \{0,1\}^{2n}$ which is assumed to be a secure cryptographic *pseudorandom generator* (PRG).[2] If $n = 128$, we can use $G(x) = \text{SHA-256}(x)$.
- **Pseudorandom ID Generation.** Every user broadcasts a sequence of IDs $\mathrm{id}_1, \mathrm{id}_2, \ldots \in \{0,1\}^n$. The i-th id_i is broadcast at any time in the window $[t_0 + \mathrm{dt} \cdot (i-1), t_0 + \mathrm{dt} \cdot i[$, where t_0 is the start time. To generate these IDs, the user initially samples a random n-bit seed S_0, and then computes
$$(S_i, \mathrm{id}_i) \leftarrow G(S_{i-1})$$
for $i = 1, 2, \ldots$. After i time units, the user only stores $S^* \leftarrow S_{\max\{i-\Delta,0\}}$, the time t^* at which S^* was generated, the current S_i, and the time t_i at which S_i was generated. Note that if the device was powered off or the application disabled, we need to advance to the appropriate S_i.
- **Pseudorandom ID Collection.** For every id broadcast by a device in its proximity at time t, a user stores a pair (id, t) in its local storage $\mathcal{S}$.
- **Reporting.** To report a positive test, the user uploads $(S^*, t_{start} = t^*, t_{end} = t_i)$ to the server, which appends it to a public list $\mathcal{L}$. The server checks that t_{start} and t_{end} are reasonable before accepting the entry. Once reported, the user erases its memory and restarts the pseudorandom ID generation procedure.
- **Checking Exposure.** A user downloads $\mathcal{L}$ from the server (or the latest portion of it). For every entry $(S^*, t_{start}, t_{end})$ in $\mathcal{L}$, it generates the sequence of IDs $\mathrm{id}_1^*, \ldots, \mathrm{id}_\Delta^*$ starting from S^*, as well as estimates t_i^* of the time at which each id_i^* was initially broadcast. If $\mathcal{S}$ contains (id_i^*, t) for some $i \in \{1, \ldots, \Delta\}$ such that t and t_i^* are sufficiently close, the user is alerted of potential exposure.

Setting Delays. To prevent replay attacks, an entry $(S^*, t_{start}, t_{end})$ should be published with a slight delay. This is to prevent an id_Δ^* generated from S^* being recognized as a potential exposure by any user if immediately rebroadcast by a malicious party.

Entry Validation. Entries can (and should) be validated by attaching a signature σ on $(S^*, t_{start}, t_{end})$ when reporting, as well as (optionally) a certificate to validate this signature. An entry thus has form $(S^*, t_{start}, t_{end}, \sigma, cert)$. Entries can be validated by multiple entities, by simply re-uploading them with a new signature.

A range of designs and policies are supported by this approach. Upon an initial update, a (weakly secure) signature with an app-specific key could be attached for self-reporting. This signature does not provide any real security (as we cannot guarantee that an app-specific signing key remains secret), but can be helpful to offer improved functionality. Third-parties (like health-care providers) can re-upload an entry with their signature after validation. An app can adopt different policies on how to display a potential exposure depending on how it is validated.

We also do not specify here the infrastructure required to establish the validity of certificates, or how a user interacts with a validating party, as this is outside the scope of this description.

Fixed-Length Sequences of IDs. As stated, during the first $\Delta - 1$ time units a user will have generated a sequence of fewer than Δ IDs. During this time, the number of IDs the user has generated from its current S^* is determined by how long ago the user started the current pseudorandom ID generation procedure (either when they initially started using the protocol or when they last submitted a report). This may be undesirable information to reveal to a party that gains access to the sequence of IDs (e.g. if the user submits a report or if the party gains

[2] This means that its output, under a random input, is indistinguishable from a random string to a computationally bounded adversary.

physical access to the user's device). So to avoid revealing this information, a user may optionally iterate to S_Δ and use id_Δ as the first ID they broadcast when starting or restarting the pseudorandom ID generation procedure.

Synchronized Updates. Suppose a user updates their seed every dt amount of time after whenever they happened to originally start the ID generation process. Then it may be possible to correlate two IDs of a user by noticing that the times at which the IDs were initially broadcast were separated in time by a multiple of dt. To mitigate this it would be beneficial to have an agreed schedule of when all users update their seed. For example, if dt is 15 minutes then it might be agreed that everyone should update their seed at midnight UTC, followed by 12:15, 12:30, and so forth.

3.1.2 Privacy and Security Properties

Privacy and integrity properties of the protocol follow from the following two propositions. (Their proofs are omitted and follow from standard techniques.) In the following discussion, it is convenient to refer to an ID value id_i output by a user as *unreported* if it is not within the Δ id's generated by a seed the user has reported to the server.

Proposition 1 (Pseudorandomness): All unreported IDs are pseudorandom, i.e., no observer (different than the user) can distinguish them from random looking strings (independent from the state of the user) without compromising the security of G.

Proposition 2 (One-wayness): No attacker can produce a seed S which generates a sequence of Δ IDs that include an unreported ID generated by an honest user (not controlled by the adversary) without compromising the security of G.

To discuss the consequences of these properties on privacy and integrity, let us refer to users as either "positive" or "negative" depending on whether they decided to report as positive, by uploading their seed to the server, or not.

- *Privacy for negative users.* By the pseudorandomness property, a negative user u only broadcasts pseudorandom IDs. These IDs cannot be linked without knowledge of the internal state of u. This privacy guarantee improves with the frequency of updating the seed S_i – ideally, if a different id_i is broadcast each time, no linking is possible. This however results in less efficient checking for exposure by negative users.[3]

- *Privacy for positive users.* Upon reporting positive, the last Δ IDs generated by the positive user *can* be linked. (We discuss what this means below, and possible mitigation approaches.) However, by pseudorandomness, this is only true for the IDs generated within the infection window. Older IDs and *newer* IDs cannot be linked with those in the infection window, and with each other. Therefore, a positive user has the same guarantees as a negative user outside of the reported infection window.

- *Integrity guarantees.* It is infeasible for an attacker to upload to the server a value S^* which generates an unreported ID that equals one generated by another user. This prevents the attacker from misreporting IDs of otherwise negative users and erroneously alerting their contacts.

Timing Information and Replay Attacks. The timestamping is necessary to prevent *replay attacks*. In particular, we are concerned by adversaries rebroadcasting IDs of legitimate users (to be tested positive) outside the range of their devices. This may create a high number of false exposures to be reported.

An attack we cannot prevent is the following *relay attack*: An attacker captures an ID of an honest user at location A, sends it over the Internet to location B, where it is re-broadcast. However, as soon as there is sufficient

[3] In a Bluetooth implementation, one needs to additionally ensure that each different id_i is broadcast with a different UUID to prevent linking.

delay, the attack is prevented by maintaining sufficiently accurate timing information. (One can envision several accuracy compromises in the implementation, which we do not discuss here.)

Strong Integrity. Our integrity property does not prevent a malicious user from reporting a seed $\widetilde{S}^*$ generating an ID which has been already reported. Given an entry with seed S^*, the attacker just chooses (for example) $\widetilde{S}^*$ as the first half of $G(S^*)$. The threat of such attacks does not appear significant. However, they could be prevented with a less lightweight protocol, as we explain next. We refer to the resulting security guarantee as *strong integrity*.

Each user generates a signing/verification-key pair (sk, vk) along with the initial seed. Then, we include vk in the ID generation process, in particular let $(S_i, \mathrm{id}_i) \leftarrow G(S_{i-1}, vk)$. An entry now consists of $(S^*, t_{start}, t_{end}, vk, \sigma)$, where σ is a signature (with signing key sk) on $(S^*, t_{start}, t_{end}, vk)$. Entries with invalid signatures are ignored. (This imposes slightly stronger assumptions on G – pseudorandomness under related seeds sharing part of the input and binding of vk to S_i.)

The CEN protocol, discussed in Section 5, is the only one that targets strong integrity, though their initial implementation failed to fully achieve it. (The issue has been fixed after our report.)

3.1.3 Inference from Positive IDs

One explicit compromise we take is that *IDs of a positive user can be linked within the infection window*, and that the start and end time of the infection window is known. For example, an adversary collecting IDs at several locations can detect that the same positive user has visited several locations at which it collects broadcast identifiers. This can be abused for surveillance purposes, but arguably, surveillance itself could be achieved by other methods. The most problematic aspect is the linking of this individual with the fact that they are positive.

A natural approach to avoid linking, as in [5], is for the the server to only expose the IDs, rather than a seed from which they are computed. However, this does not make them unlinkable. Imagine, at an extreme, that the storage on the server is append only (which is a realistic assumption). Then, the IDs belonging to the same user are stored sequentially. One can obfuscate this leakage of information in several ways, for example by having the server buffer a certain amount of new IDs, and shuffle them before release. Nonetheless, the actual privacy improvement is hard to assess without a good statistical model of upload frequency. This also increases the latency of the system which directly harms its public health value.

A user could also learn at which time the exposure took place, and hence infer the identity of the positive user from other available information. We stress that the application can and should refuse to display the time of potential exposure – thus preventing a "casual attacker" from learning timing information. However, a malicious app can always remember at which time an ID has been seen.

3.2 Mobile-Assisted Contact Tracing Interviews

Contact tracing interviews are laborious and often miss important events due to the limitations of human memory. Our plan to assist here is to provide information to the end user that can (with consent) be shared with a public health organization charged with performing contact tracing interviews. This is not an exposure of the entire observational log, but rather an extract of the information which is requested in a standard contact tracing interview. We have been working with healthcare teams from Boston and the University of Washington on formats and content of information that are traditionally sought by public health agencies. Ideally, such extraction can be done working with the user before a contact tracing interview even occurs to speed the process.

3.3 Narrowcasting

Healthcare authorities from NYC have informed us that they would love to have the ability to make public service announcements which are highly tailored to a location or to a subset of people who may have been in a certain

region during specific periods of time. This capability can be enabled with a public server supporting (area x time,message) pairs. Here "area" is a location, a radius (minimum 10 meters), a beginning time and an ending time. Only announcements from recognized public health authorities are allowed.

Anyone can manually query the public server to determine if there are messages potentially relevant to them per their locations and dwells at the locations over a period of time. However, simple automation can be extremely helpful as phones can listen in and alert based on filters that are dynamically set up based on privately-held locations and activities. Upon downloading (area x time, message) pairs a phone app (for example) can automatically check whether the message is relevant to the user. If it is relevant, a message is relayed to the device owner.

Querying the public server provides *no* information to the server through the protocol itself, because only a simple copy is required.

4 Alternative Approaches

We discuss some alternative approaches to mobile tracing. Some of these are expected to be adopted in existing and future contact-tracing proposals, and we discuss them here.

Hart et al. [10] provides a useful high-level understanding of the issues involved in contact tracing. They discuss, among other topics, the value of using digital technology to scale contract tracing and the trade-offs between different classes of solutions.

4.1 Reporting collected IDs

PACT users upload their locally *generated* IDs upon a positive report. An alternative is to upload *collected* IDs of potentially at risk users. This approach (which we refer to as the *dual* approach) has at least one clear security disadvantage and one mild privacy advantage over PACT. (The latter is only true if the system is carefully implemented, as we explain below.)

Disadvantages: Reliability and Integrity Attacks. In the dual approach, a malicious user cannot be prevented from very easily reporting a very large number of IDs which were not generated by users in physical proximity. These IDs could have been collected by colluding parties elsewhere, at any time before the report. Such attacks can seriously hurt the reliability of the system. In PACT, to achieve a similar effect, the attacker needs to (almost) simultaneously broadcast the same ID in direct proximity of all individuals who should be falsely alerted to be potentially at risk.

PACT ensures integrity of positive reporting by exhibiting a seed generating these IDs, known only to the reporter. A user u cannot frame another negative user u' as a positive user by including an ID generated by u'. In the dual approach, user u' could be framed for example by uploading IDs that have been broadcast in their surroundings.

Advantage: Improved Temporal Ambiguity. Both in the dual approach and in PACT-like designs, a user at risk can de-anonymize a positive user from the time at which the matching ID was generated/collected, and other contextual information (e.g., a surveillance video).

The dual approach offers a mitigation to this using *re-randomization* of IDs. We explain one approach [12]. Let $\mathbb{G}$ be a prime-order cyclic group with generator g (instantiated via a suitable elliptic curve).

1. Each user u chooses a secret key s_u as a random element in $\mathbb{Z}_p$.
2. Each broadcast ID takes the form $\mathrm{id}_i = (g^{r_i}, g^{r_i s_u})$, where $r_1, r_2, \ldots$ are random elements of $\mathbb{Z}_p$.
3. To upload an ID with form $\mathrm{id} = (x, y)$ with a report, a positive user uploads instead a re-randomized version $\mathrm{id}' = (x^r, y^r)$, where r is a fresh random value from $\mathbb{Z}_p$.

4. To determine whether they are at risk, user u checks whether an ID of the form $\mathrm{id} = (x, y)$ such that $y = x^{s_u}$ is stored on the server.

Under a standard cryptographic assumption – the so-called *Decisional Diffie-Hellman* (DDH) assumption – the IDs are pseudorandom. Further, a negative user who learns they are at risk cannot tell which one of the IDs they broadcast has been reported, as long as the reporting user re-randomized them and all IDs have been generated using the same s_u. Note that incorrect randomization only hurts the positive user.

Crucially, however, the privacy benefit inherently relies on each user u re-using the same s_u, and we cannot force a malicious user to comply. For example, to track movements of positive users, a surveillance entity can generate IDs at different locations with form (x, y) where $y = x^{s_L}$ and s_L depends on the location L. Identifiers on the server with form (x, x^{s_L}) can then be traced back to location L. A functionally equivalent attack is in fact more expensive against PACT, as this would require storing all IDs of users broadcast at location L.

4.2 Centralized (Trusted Third-Party) Approaches

We discuss an alternative *centralized* approach here, which relies on a trusted third party (TTP), typically an agency of a government. Such a solution requires an initial *registration phase* with the TTP, where each user subscribes to the service. Moreover, the protocol operates as follows:

1. Users broadcast random-looking IDs and gather IDs collected in their proximity.

2. Upon a positive test, a user reports to the TTP all of the IDs collected in their proximity during the relevant infection window. The TTP then alerts the users who generated these IDs, who are now at risk.

In order for the TTP to alert potentially at risk users, it needs to be able to identify the owners of these identifiers. There a few technical solutions to this problem.

- One option is to have the TTP generate all IDs which are used by the users - this requires either storing them or (in case only seeds generating them are stored) a very expensive check to identity at risk users.

- A more efficient alternative for the TTP (but with larger identifiers) goes as follows. The trusted third-party generates a public-key/secret-key pair (sk, pk), making pk public. It also gives a unique token τ_u to each user u upon registration, which it remembers. Then, the i-th ID of user u is $\mathrm{id}_i = \mathrm{Enc}(pk, \tau_u)$. (Note that encryption is randomized here, so every id_i appears independent from prior ones.) The TTP can then efficiently identify the user who generated id_i by decrypting it.

Privacy Considerations. Such a centralized solution offers better privacy against attackers who do not collude with the TTP - in particular, only pseudorandom identifiers are broadcast all times. Moreover, at risk individuals only learn that one of the IDs they collected belongs to a positive individual. A -risk users can still collude, learning some information from the time of being reported at risk, and correlate identifiers belonging to the same positive user, but this is harder.

The biggest drawback of this solution, however, is the high degree of trust on the TTP. For example:

- The TTP learns the identities of all at risk users who have been in proximity of the positive subject.

- The TTP can, at any time and independently of any actual report, learn the identity of the user u who broadcasts a particular ID, or at least link them to their token τ_u. This could be easily exploited for surveillance of users adopting the service.

Security Consideration. As in the dual approaches described above, it is trivial for a malicious party identifying as honest to report valid identifiers of other users (which may have been collected in a distributed

fashion) to erroneously alert them as being at risk. Replay attacks can be mitigated by encrypting extra meta-data along with τ_u (e.g., a timestamp), but this would make IDs even longer.

If the TTP is malicious it can target specific users to falsely claim they are at risk or to refrain from informing them when they actually are at risk.

4.3 Absolute-Location–Centric Mobile Tracing Methods

It is also possible to design protocols based on the sensing of absolute locations (GPS, and GPS extended with dead reckoning, wifi, other signals per current localization methods) consistent with "If you do not report as being positive, then no information of yours will leave your phone" (see Section 2). For example, a system could upload location traces of positives (cryptographically, in a secure manner), and then negative users, whose traces are stored on their phones could intersect their traces with the positive traces to check for exposure. This could potentially be done with stronger cryptographic methods to limit the exposure of information about these traces to negative users; one could think of this as a more general version of *private-set intersection* (PSI) [1, 9, 16]. However, such solutions would still reveal traces of positives to a server.

There are two reasons why we do not focus on the details of such an approach here:

- Current localization technologies are not as accurate as the use of Bluetooth-based proximity detection, and may not be accurate enough to be consistent with medically suggested definitions for exposure.
- Approaches employing the sensing and collection of absolute location information would need to rely more heavily on cryptographic protocols to keep the positive users traces secure.

However, this is an approach worth keeping in mind as an alternative, per assessments of achievable accuracies and relevance of the latter accuracies for public health applications.

5 Related Efforts

There are an increasing number of contact tracing applications being created with different protocols. We will briefly discuss a few of these and how their mobile tracing protocols compare with the approaches described in Section 3.1 and 4.

5.1 Similar Bluetooth-Based Efforts

The privacy-sensitive mobile tracing protocols proposed by CoEpi [6], CovidWatch [1], as well as $\mathrm{DP}^3\mathrm{T}$ [3], have a similar structure to our proposed protocol. We briefly describe the technical differences between all of these protocols and discuss the implications of these differences.

Similar to our proposed protocol, these are based on producing pseudorandom IDs by iteratively applying a PRG G to a seed. CoEpi and CovidWatch use the Contact Event Numbers (CEN) protocol, in which the initial seed is derived from a digital signature signing key rak and G is constructed from two hash functions (which during each iteration incorporate an encoding of the number of iterations done so far and the verification key rvk which matches rak). Another proposal is the $\mathrm{DP}^3\mathrm{T}$ [3] protocol, in which G is constructed from a hash function, a PRF, and another PRG. The latter PRG is used so that a single iteration of G produces all the IDs needed for a day. These IDs are used in a random order throughout the day. Both of these (under appropriate cryptographic assumptions) achieve the same sort of pseudorandomness and one-wayness properties as our protocol.

The incorporation of rvk into G with CEN is intended to provide strong integrity and allow a reporting user to include a memo with their report that is cryptographically bound to the report. Two ideas for what such a memo might include are a summary of the user's self-reported symptoms (CoEpi) or an attestation from a third party verifying that the user tested positive (CovidWatch). Because a counter of how many times the seed has

been updated is incorporated into G, a report must specify the corresponding counters. This leaks how long ago the user generated the initial seed, which could potentially be correlated with identifying information about the user (e.g., when they initially downloaded the app).

An earlier version of CEN incorrectly bound the digital signature key to the identifiers in a report. Suppose an honest user has submitted a report for id_j through $id_{j'}$ (for $j < j'$) with a user chosen memo. Given this report, an attacker could create their own report that verifies as valid, but includes the honest user's id_i for some i between j and j' together with a memo of the attacker's choosing. A fix was proposed after we contacted the team behind the CEN protocol.

The random order of a user's IDs for a day by $\mathrm{DP}^3\mathrm{T}$ is intended to make it difficult for an at risk individual to identify specifically when they were at risk (and thus potentially, by whom they were exposed). A protocol cannot hope to hide this sort of timing information from an attacker that chooses to record the time when they received every ID they see; this serves instead as a mitigation against a casual attacker using an app that does not store this sort of timing information. In our protocol and CEN, information about the exposure time is not intended to be as hidden at the protocol. In our protocol the time an ID was used is even included as part of a report and used to prevent replay attacks, as discussed earlier. CEN does not use timing information to prevent replay attacks, but considers that an app may choose to give users precise information about where they were exposed (so the user can reason about how likely this potential exposure was to be an actual exposure).

A similar protocol idea was presented in [5]. It differs from the aforementioned proposals in that individual IDs are uploaded to the server, rather than a seed generating them (leading to increased bandwidth and storage). Alternatives using bloom filters to reduce storage are discussed, but these inherently decrease the reliability of the system. $\mathrm{DP}^3\mathrm{T}$ also recently included a similar protocol as an additional option, using cuckoo filters in place of bloom filters.

5.2 Centralized Example

The TraceTogether [20] app is currently deployed in Singapore. It uses the BlueTrace protocol designed by at team at the Government Technology Agency of Singapore. This protocol is closely related to the encryption-based technique discussed in Section 4.2.

5.3 Absolute-Location–Centric Example

The Private Kit: Safe Paths app [19, 17] intends to use an absolute-location–centric approach to mobile tracing. They intend to mitigate some of the downsides discussed in Section 4.3 by reported location traces of positive users to be partially redacted. It is unclear what methodology they intend to use for deciding how to redact traces. The trade-off in this redaction process between how easily a positive user can be identified from their trace and how much information must be removed from it (decreasing its usefulness). They intend to use cryptographic protocols (likely based on [2]) to minimize the amount of information revealed about positive users' traces.

5.4 Other Efforts

A group of scientists at the Big Data Institute of Oxford University have proposed the use of a mobile contact-tracing app [13, 4] based on their analysis in [8]. The NextTrace [14] project aims to coordinate with COVID-19 testing labs and users, providing software to enable contact tracing. The details of these proposals and the privacy protections they intend to provide are not publicly available.

The projects we refer to are only a small selection of the mobile contract-tracing efforts currently underway. A more extensive listing of these projects is being maintained at [22], along with other information of interest to contract tracing.

6 Discussion and Further Considerations

6.1 Interoperability of Different Protocols

Most protocols like ours store a seed on a server, which is then used to deterministically generate a sequence of identifiers. Details differ in how exactly these sequences are generated (including the adopted cryptographic algorithms). However, it appears relatively straightforward for apps to be modified to support all of these different sequence formats. A potential challenge is data from different protocol may provide different levels of protection (e.g., the lack of timing information may reduce the effectiveness against replay attacks). This difference in reliability may be surfaced via the user-interface.

In order to support multiple apps accessing servers for different services, it is important to adopt an interoperable format for entries to be stored on a server and possibly, to develop a common API.

6.2 Ethics Considerations

We acknowledge that ethical questions arise with contact tracing and in the development and adoption of any new technology. The question of how to balance what is revealed for the good of public health vs individual freedoms is one that is central to public health law. We iterate that privacy is already impacted by tracing practices. In some nations, positively tested citizens are required, either by public health policy or by law, to disclose aspects of their history. Such actions and laws frame multiple concerns about privacy and freedom, and bring up important questions. The purpose of this document is lay out some of the technological capabilities, which supports broader discussion and debate about civil liberties and the risks that contact tracing can pose to civil liberties.

Another concern is accessibility to the service: not everyone has a phone (or will have the service installed). One consequence of this is that the quality of contract tracing in a certain population inherently depends on factors orthogonal to the technological aspects, which in turn raises important questions about fairness.

6.3 Larger Considerations of Testing, Tracing, and Timeouts

Tracing is one part of a conventional epidemic response strategy, based on Tests, Tracing, and Timeouts (TTT). Programs involving all three components are as follows:

- Test heavily for the virus. South Korea ran over 20 tests per person found with the virus.
- Trace the recent physical contacts for anyone who tests positive. South Korea conducted *mobile contact tracing* using telecom information.
- Timeout the virus by quarantining contacts until their immune system purges the virus, rendering them non-infectious.

The mobile tracing approach allows this strategy to be applied at a dramatically larger scale than only relying on human contact tracers.

6.4 Challenge of Wide-Scale Adoption

This chain is only as strong as its weakest link. Widespread testing is required and wide-scale adoption must occur. Furthermore, strategies must also be employed so that citizens takes steps to self-quarantine or seek testing (as indicated) when they are exposed. We cannot assume 100 percent usage of the application and concomitant enlistment in TTT programs. Studies are needed of the efficacy of the sensitivity of the effectiveness of the approach to different levels of subscription in a population.

Acknowledgments

The authors thank Yael Kalai for numerous helpful discussions, along with suggesting the protocol outlined in Section 4.1. We also thank Ramesh Raskar and the Microsoft Cryptography group including Kristin Lauter, Kim Laine, Esha Ghosh, and Melissa Chase for private set intersection discussions related to locations. We thank Edward Jezierski, Nicolas di Tada, Vi Hart, Ivan Evtimov, and Nirvan Tyagi for numerous helpful discussions. We also graciously thank M Eifler for designing all the figures. Sham Kakade acknowledges funding from the Washington Research Foundation for Innovation in Data-intensive Discovery, the ONR award N00014-18-1-2247, NSF grants #CCF-1637360 and #CCF 1740551. Jacob Sunshine acknowledges funding from NIH (K23DA046686) and NSF (1914873, 1812559). Stefano Tessaro acknowledges support from a Sloan Research Fellowship and from the NSF under grants CNS-1553758, CNS-1719146.

References

[1] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, page 86–97, New York, NY, USA, 2003. Association for Computing Machinery.

[2] A. Berke, M. Bakker, P. Vepakomma, R. Raskar, K. Larson, and A. S. Pentland. Assessing disease exposure risk with location histories and protecting privacy: A cryptographic approach in response to a global pandemic, 2020.

[3] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. *Journal of cryptographic engineering*, 2(2):77–89, 2012.

[4] D. Bonsall, M. Parker, and C. Fraser. Sustainable containment of covid-19 using smartphones in china: Scientific and ethical underpinnings for implementation of similar approaches in other settings. `https://github.com/BDI-pathogens/covid-19_instant_tracing`. Accessed: 2020-04-05.

[5] R. Canetti, A. Trachtenberg, and M. Varia. Anonymous collocation discovery:taming the coronavirus while preserving privacy, 2020. Accessed: 2020-04-05.

[6] Coepi: Community epidemiology in action. `https://www.coepi.org/`. Accessed: 2020-04-05.

[7] Covid watch. `https://www.covid-watch.org/`. Accessed: 2020-04-05.

[8] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, M. Parker, D. Bonsall, and C. Fraser. Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. *Science*, 2020.

[9] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In C. Cachin and J. L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 1–19, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[10] V. Hart, D. Siddarth, B. Cantrell, P. E. Lila Tretikov, J. Langford, S. Leibrand, S. Kakade, S. Latta, D. Lewis, S. Tessaro, and G. Weyl. Outpacing the virus: Digital response to containing the spread of covid-19 while mitigating privacy risks. `https://ethics.harvard.edu/outpacing-virus`. Accessed: 2020-04-05.

[11] S. Josefsson and I. Liusvaara. Rfc 8032: Edwards-curve digital signature algorithm (eddsa). *Internet Engineering Task Force (IETF)*, 2017.

[12] Y. Kalai. Personal Communication.

[13] Mobile app - coronavirus fraser group. https://045.medsci.ox.ac.uk/mobile-app. Accessed: 2020-04-05.

[14] Nexttrace. https://nexttrace.org/. Accessed: 2020-04-05.

[15] K. Pietrzak. Delayed authentication: Replay and relay attacks on dp-3t. Cryptology ePrint Archive, Report 2020/418, 2020. https://eprint.iacr.org/2020/418.

[16] B. Pinkas, T. Schneider, G. Segev, and M. Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 515–530, Washington, D.C., Aug. 2015. USENIX Association.

[17] R. Raskar, I. Schunemann, R. Barbar, K. Vilcans, J. Gray, P. Vepakomma, S. Kapa, A. Nuzzo, R. Gupta, A. Berke, D. Greenwood, C. Keegan, S. Kanaparti, R. Beaudry, D. Stansbury, B. B. Arcila, R. Kanaparti, V. Pamplona, F. M. Benedetti, A. Clough, R. Das, K. Jain, K. Louisy, G. Nadeau, V. Pamplona, S. Penrod, Y. Rajaee, A. Singh, G. Storm, and J. Werner. Apps gone rogue: Maintaining personal privacy in an epidemic, 2020.

[18] E. Rescorla. Rfc 8446: The transport layer security (tls) protocol version 1.3. *Internet Engineering Task Force (IETF)*, 2018.

[19] Private kit: Safe paths; privacy-by-design contact tracing. http://safepaths.mit.edu/. Accessed: 2020-04-05.

[20] Tracetogether. https://www.tracetogether.gov.sg/. Accessed: 2020-04-05.

[21] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, L. Barman, S. Chatel, K. Paterson, S. Capkun, D. Basin, D. Jackson, B. Preneel, N. Smart, D. Singelee, A. Abidin, S. Guerses, M. Veale, C. Cremers, R. Binns, and T. Wiegand. Decentralized privacy-preserving proximity tracing. https://github.com/DP-3T/documents/blob/master/DP3T%20White%20Paper.pdf. Accessed: 2020-04-05.

[22] Unified research on privacy-preserving contact tracing and exposure notification for covid-19. https://bit.ly/virustrackertracker. Accessed: 2020-04-05.

[23] R. White and E. Horvitz. From web search to healthcare utilization: Privacy-sensitive studies from mobile data. *JAMIA: Journal of the American Medical Informatics*, 2012.

Appendix

A Issues around Practical Implementation

A number of practical issues and details may arise with implementation.

1. With regards to anonymity, if the protocol is implemented over the internet, then GeoIP lookups can be used to localize the query-maker to a varying extent. People who really care about this could potentially query through an anonymization service.

2. The narrowcast messages in particular may be best expressed through existing software map technology. For example, we could imagine a map querying the server on behalf of users and displaying public health messages on the map.

3. The bandwidth and compute usage of a phone querying the full database may be too high. To avoid this, it's reasonably easy to augment the protocol to allow users to query within a (still large) region. We mention one such approach below.

4. Disjoint authorities. Across the world, there may be many testing authorities which do not agree on a common infrastructure but which do wan to use the protocol. This can be accommodated by enabling the phone app to connect to multiple servers.

5. The mobile proximity tracing does not directly inform public authorities who may be a contact. However, it does provide some bulk information, simply due to the number of posted messages.

There are several ways to implement the server. A simple approach, which works fine for not-too-many messages just uses a public GitHub repository.

A more complex approach supporting regional queries is defined next.

A.1 Regional Query Support

Anyone can ask for a set of messages relevant to some region R where R is defined by a latitude/longitude range with messages after some timestamp. More specific subscriptions can be constructed on the fly based on policies that consider a region R and privately observed periods of time that an individual has spent in a region. Such scoped queries and messaging services that relay content based on location or on location and periods of time are a convenience to make computation and communication tractable. The reference implementation uses regions greater in size than typical GeoIP tables.

To be specific, let's first define some concepts.

- Region: A region consists of a latitude prefix, a longitude prefix, and the precision in each. For example, New York which is at 40.71455 N, -74.00712 E can be coarsened to 40 N, -74 E with two digits of precision (the actual implementation would use bits).

- Time: A timestamp is specified in the number of seconds (as a 64 bit integer) since the January 1, 1970.

- Location: A location consists of a full precision Latitude and Longitude

- Area: An area consists of a Location, a Radius, a beginning Time, and an ending Time.

- Bluetooth Message: A Bluetooth message consists of a fixed-length string of bytes. It is used with the Bluetooth sensory log to discover if there is a match, which results in a warning that the user may have been in contact with an infected person.

- Message: A message is a cryptographically signed string of bytes which is interpreted by the phone app. This is used for either a public health message (announced to the user if the sensory log matches) or a Bluetooth Message.

With the above defined, there are two common queries that the server supports as well as an announcement mechanism.

- GetMessages(Region, Time) returns all of the (Area, Message) pairs that the server has added since Time for the Region. The app can then check locally whether the Area intersects with the recorded sensory log of (Location,Time) pairs on the phone, and alert the user with the Message if so.
- HowBig(Region, Time) returns the (approximate) number of bytes worth of messages that would be downloaded on a GetMessages call with the same arguments. HowBig allows the phone app to control how much information it reveals to the server about locations/times of interest according to a bandwidth/privacy tradeoff. For example, the phone could start with a very coarse region, specifying higher precision regions until the bandwidth required is acceptable, then invoke GetMessages. (This functionality is designed to support controlled anonymity across widely varying population densities.)
- Announce(Area,Message) uploads an (Area, Message) pair for general distribution. To prevent spamming, the signature of the message is checked against a whitelist defined with the server.

B Alternative Protocol

We propose an alternative to the protocol in Section 3.1. One main difference is that the server *cannot* generate the IDs broadcast by a positive user, and only stores a short verification key used to identify IDs broadcast by the positive user. While this does not prevent many of the inference scenarios we discussed above, this appears to be a desirable property. As we explain below, this protocol offers a different cost for checking exposure, which may be advantageous in some deployment scenarios.

This alternative approach inherently introduces risks of replay attacks which cannot be prevented by storing timestamps, because the server obtains no information about the times at which IDs have been broadcast. To overcome this, we build on top of a very recent approach of Pietrzak [15] for replay-attack protection. (Along similar lines, this can also be extended to relay-attack protection by including GPS coordinates, but we do not describe this variant here.)

- **Setup and Parameters.** We fix an understood time unit dt. We make use of a *digital signature scheme* specifying algorithms for key generation, signing, and verification, denoted Kg, Sign, and Vrfy, respectively. We also use a hash function $H : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$. We can use $H(k, x) = \mathrm{SHA256}(k||x)$ and Ed25519 [3, 11] for the signature scheme.
- **Pseudorandom ID Generation.** Every user broadcasts a sequence of IDs $\mathrm{id}_{1,d}, \mathrm{id}_{2,d}, \ldots$ during day d. The i-th $\mathrm{id}_{i,d}$ is broadcast at any time in the window $[t_d + \mathrm{dt} \cdot (i-1), t_d + \mathrm{dt} \cdot i[$, where t_d is the time at the beginning of the day. To generate these IDs, the user runs Kg to generate a daily signing key / verification key pair $(\mathrm{sk}_d, \mathrm{vk}_d)$, and keeps both of them secret. They also determine the current time $t_i = t_d + \mathrm{dt} \cdot (i-1)$. Finally, the user samples n-bit random strings R_i and r_i and computes the identifier as

  $$\mathrm{id}_i = (\sigma_i, R_i, h_i),$$

  where $\sigma_i = \mathsf{Sign}(\mathrm{sk}_d, R_i||h_i)$ and $h_i = H(r_i, t_i)$. They broadcast $(\mathrm{id}_i, r_i, t_i)$. When day d ends the user deletes their signing key sk_d. (The verification key vk_d is *not* deleted, until an amount of time equal to the infection window has elapsed.)

- **Pseudorandom ID Collection.** For every $(\mathrm{id}_i = (\sigma_i, R_i, h_i), r_i, t_i)$ broadcast by a device in their proximity, a user first checks if t_i is sufficiently close to their current time and if $h_i = H(r_i, t_i)$. If so, they store id_i in their local storage $\mathcal{S}$.
- **Reporting.** To report a positive test, the user uploads each of their recent vk_d to the server, which appends them to a public list $\mathcal{L}$. Once reported, the user erases their memory and restarts the pseudorandom ID generation procedure.
- **Checking Exposure.** A user downloads $\mathcal{L}$ from the server (or the latest portion of it). For every entry vk in $\mathcal{L}$ and every entry (σ, R, h) in $\mathcal{S}$, they run $\mathsf{Vrfy}(\mathrm{vk}, \sigma, R||h)$. If this returns true, the user is alerted of potential exposure.

Efficiency Comparisons. Let Δ be the number of IDs broadcast over the infection window. Let $S = |\mathcal{S}|$ be the size of the local storage. Let L be the number of new verification keys a user downloads. To check exposure, the protocol from Section 3.1 roughly runs in time

$$L \times \Delta \times \log(S) \times t_G \ ,$$

where t_G is the time needed to evaluate G. In contrast, for the protocol in this section, the time is

$$L \times S \times t_{\mathsf{Vrfy}} \ ,$$

where t_{Vrfy} is the time to verify a signature. One should note that t_{Vrfy} is generally larger than t_G, but can still be quite fast. (For example, Ed25519 enables fast batch signature verification.)

Therefore, the usage of this scheme makes particular sense if a user does not collect many IDs, i.e., S is small relative to $\Delta \cdot \log(S)$.

Another important point of comparison is how many bits need to be broadcast by the protocol.

Assumptions. We require the following two standard properties for the hash function H:

- **Pseudorandomess:** For any x and a randomly chosen $r \in \{0,1\}^n$, the output $H(r, x)$ looks random to anyone that doesn't know r.
- **Collision resistance:** It is hard to find distinct inputs to H that produce the same output.

Of our digital signature scheme we require the following three properties. The first is a standard property of digital signature schemes. The latter two are not commonly required of a digital signature scheme, so one needs to be careful when choosing a signature scheme to implement this protocol. We have verified that these properties are achieved by Ed25519 under reasonable cryptographic assumptions.

- **Unforgeability:** Given vk and examples of $\sigma = \mathsf{Sign}(\mathrm{sk}, m)$ for attacker-chosen m, an attack cannot produce a new (σ', m') for which $\mathsf{Vrfy}(\mathrm{vk}, \sigma', m')$ returns true.
- **One-wayness:** Given examples of $\sigma = \mathsf{Sign}(\mathrm{sk}, m)$ for attacker-chosen m (but not given vk), an attacker cannot find vk' for which $\mathsf{Vrfy}(\mathrm{vk}', \sigma, m)$ returns true for any of the example (σ, m).
- **Pseudorandomess:** The output of $\mathsf{Sign}(\mathrm{sk}, \cdot)$ looks random to an attacker that does not know vk or sk.

Privacy and Security Properties. We discuss the privacy and integrity properties this protocol has in common with the earlier protocol, as well as some newer properties not achieved by the earlier protocol.

- *Privacy for negative users.* By the pseudorandomness property, the signatures broadcast by a user u look pseudorandom. Beyond that, u broadcasts two random strings and their view of the current time t_i which

is already known by any device hearing the broadcast.[4] Thus these broadcasts cannot be linked without knowledge of the internal state of u. As before, this privacy guarantee improves with the frequency of generating new IDs.

- *Privacy for positive users.* Upon reporting positive, the IDs broadcast by a user within a single day *can* be linked to each other. IDs broadcast on different days can be linked if the server does not hide which vk's were reported together. Older IDs from days before the infection window and *newer* IDs from after the report cannot be linked with those in the infection window or with each other. Therefore, a positive user has the same guarantees as a negative user outside of the reported infection window.

- *Integrity guarantees.* It is infeasible for an attacker to upload to the server a value vk which verifies an unreported ID that was broadcast by another user. This prevents the attacker from misreporting IDs of otherwise negative users and erroneously alerting their contacts.

- *Replay protection.* The incorporation of t in each ID prevents an attacker from performing a replay attack where they gather IDs of legitimate users (to be tested positive) and re-broadcast the IDs at a later time to cause false beliefs of exposure. A vk reported to the server cannot be used to broadcast further IDs that will be recognized by other users as matching that report.

- *Non-sensitive storage.* Because $H(r_i, t_i)$ looks random, the information intentionally stored by the app together with an ID does not reveal when the corresponding interaction occurred. (Of course, it may be possible to infer information about t_i through close examination of how the ID was stored, e.g., where it was written in memory as compared to other IDs.)

[4] We note that this information *does* have the potential to be used for tracking if a user's device has some large systematic bias in its measurement of time.

Decentralized Privacy-Preserving Proximity Tracing

Carmela Troncoso[1], Mathias Payer[1], Jean-Pierre Hubaux[1], Marcel Salathé[1], James Larus[1], Wouter Lueks[1], Theresa Stadler[1], Apostolos Pyrgelis[1], Daniele Antonioli[1], Ludovic Barman[1], Sylvain Chatel[1], Kenneth Paterson[2], Srdjan Capkun[2], David Basin[2], Jan Beutel[2], Dennis Jackson[2], Marc Roeschlin[2], Patrick Leu[2], Bart Preneel[3], Nigel Smart[3], Aysajan Abidin[3], Seda Gürses[4], Michael Veale[5], Cas Cremers[6], Michael Backes[6], Nils Ole Tippenhauer[6], Reuben Binns[7], Ciro Cattuto[8], Alain Barrat[9], Dario Fiore[10], Manuel Barbosa[11], Rui Oliveira[11], and José Pereira[11]

[1]EPFL
[2]ETH Zurich
[3]KU Leuven
[4]TU Delft
[5]University College London
[6]CISPA Helmholtz Center for Information Security
[7]University of Oxford
[8]University of Torino / ISI Foundation
[9]Aix Marseille Univ, Université de Toulon, CNRS, CPT
[10]IMDEA Software Institute
[11]INESC TEC / FCUP
[11]INESC TEC / UMinho

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Executive Summary

This document describes and analyzes a system for secure and privacy-preserving proximity tracing at large scale. This system provides a technological foundation to help slow the spread of SARS-CoV-2 by simplifying and accelerating the process of notifying people who might have been exposed to the virus so that they can take appropriate measures to break its transmission chain. The system aims to minimise privacy and security risks for individuals and communities and guarantee the highest level of data protection.

The goal of our proximity tracing system is to determine who has been in close physical proximity to a COVID-19 positive person and thus exposed to the virus, without revealing the contact's identity or where the contact occurred. To achieve this goal, users run a smartphone app that continually broadcasts an ephemeral, pseudo-random ID representing the user's phone and also records the pseudo-random IDs observed from smartphones in close proximity. When a patient is diagnosed with COVID-19, she can upload pseudo-random IDs previously broadcast from her phone to a central server. Prior to the upload, all data remains exclusively on the user's phone. Other users' apps can use data from the server to locally estimate whether the device's owner was exposed to the virus through close-range physical proximity to a COVID-19 positive person who has uploaded their data. In case the app detects a high risk, it will inform the user.

The system provides the following security and privacy protections:

- **Ensures data minimization**. The central server only observes anonymous identifiers of COVID-19 positive users without any proximity information. Health authorities learn no information except that provided when a user reaches out to them after being notified.

- **Prevents abuse of data**. As the central server receives the minimum amount of information tailored to its requirements, it can neither misuse the collected data for other purposes, nor can it be coerced or subpoenaed to make other data available.

- **Prevents tracking of users.** No entity can track users that have *not* reported a positive diagnosis. Depending on the implementation chosen, others can only track COVID-19 positive users in a small geographical region limited by their capability to deploy infrastructure that can receive broadcasted Bluetooth beacons.

- **Graceful dismantling.** The system will dismantle itself after the end of the epidemic. COVID-19 positive users will stop uploading their data to the central server, and people will stop using the app. Data on the server and in the apps is removed after 14 days.

We are publishing this document to inform the discussion revolving around the design and implementation of proximity tracing systems. This document is accompanied by other documents containing an overview of the data protection compliance of the design, an extensive privacy and security risk evaluation of digital proximity tracing systems, a proposal for interoperability of multiple systems deployed in different geographical regions, and alternatives for developing secure upload authorisation mechanisms.

A Need, purpose and requirements

In this section, we describe the problems that motivate the need for digital proximity tracing systems, the purpose of our system, and its requirements. We discuss additional, potentially desirable goals that have been proposed for digital proximity tracing that our design does not attempt to achieve.

In the next sections, we present three protocols to implement decentralized proximity tracing. One protocol is an extremely lightweight system that permits limited tracing of COVID-19 positive users under very specific conditions. The other protocols provide additional privacy properties at a moderate increase in cost.

A.1 Context and need

Beyond its medical and economic consequences, the COVID-19 pandemic poses a severe challenge to healthcare authorities and governments: how to contain the spread of the SARS-CoV-2 virus while simultaneously returning to normality. There has been a vigorous debate about the best strategy to achieve these two goals. However, many experts advocate for a strategy based on testing, contact tracing, isolation, and quarantine[1] (TTIQ - see also the contact tracing[2] and proximity tracing[3] policy briefs by the Swiss National COVID-19 Science Task Force).

A cornerstone of the TTIQ strategy is effective contact tracing. Contact tracing **identifies individuals who have been exposed** to a COVID-19 positive person and consequently are at risk of having contracted COVID-19. Identifying the contacts of a confirmed positive case, so they can go into quarantine as quickly as possible**,** is of crucial importance to successfully containing the spread of the virus. In particular, presymptomatic transmission - i.e., transmission during the 2-3 days before the onset of symptoms - is estimated to account for about half of the overall transmission.[4] Thus, asking all exposed contacts to go into quarantine very rapidly is key in breaking the transmission chains of the virus.

Manual contact tracing relies on interviews conducted by trained personnel. This process alone is limited in responding to the demands of COVID-19 for two reasons:

1) In-person or over-the-phone contact tracing interviews are time consuming, and in order to handle a large number of infected people, they require many trained contact tracers and are therefore difficult to scale rapidly.

2) Even with a long, in-depth interview, the list of contacts from the interview is often incomplete. In the case of a respiratory disease, such as COVID-19, any person who has been in close physical proximity to a COVID-19 positive person should be listed as a contact. This includes strangers that a diagnosed person will not be able to recall or identify in an interview, such as nearby passengers on public transportation. Even remembering all acquaintances one has encountered over the past two weeks can be a challenge.

These issues have prompted numerous initiatives towards the use of digital proximity tracing systems to support human contact tracers.

[1] Salathé M et al. COVID-19 epidemic in Switzerland: on the importance of testing, contact tracing and isolation. *Swiss Med Wkly.* March 19, 2020

[2] Swiss National COVID-19 Science Task Force. "SARS-CoV-2 contact tracing strategy: epidemiologic and strategic considerations" (26 April 2020) Accessed on 23 May 2020: `https://ncs-tf.ch/en/policy-briefs/contact-tracing-strategy-26-april-20-en/download`

[3] National COVID-19 Science Task Force. "Digital Proximity Tracing" (15 May 2020) `https://ncs-tf.ch/en/policy-briefs/digital-proximity-tracing-15-may-20-en/download`

[4] See e.g. He X et al. Temporal dynamics in viral shedding and transmissibility of COVID-19. Nature Medicine 2020; Ganyani T et al. Estimating the generation interval for coronavirus disease (COVID-19) based on symptom onset data, March 2020. Eurosurveillance 2020.

A.2 Purpose

The primary purpose of digital proximity tracing systems is to provide a **mechanism that alerts users who have been in close physical proximity to a confirmed COVID-19 positive case for a prolonged duration** that they may have been exposed to the virus. Exposure to the virus does not imply that the person has contracted COVID-19, but serves as a trigger for a precautionary intervention recommended by a public health authority, such as testing or quarantine. This process does not require revealing the identity of the diagnosed person or when and where the contact occurred.

Most adults carry smartphones throughout the day. This opens the possibility of a mobile application that collects data about close physical proximity between individuals and thus allows the tracing of contacts that might have been infected through droplet transmission, widely assumed to be the dominant transmission route of SARS-CoV-2[5]. To this end, the application records exposure events between personal smartphones. An **exposure event** is recorded when two phones are in close physical proximity for a period of time, for some pre-defined value for distance and duration. **Proximity tracing** is the process that the app uses to calculate whom to notify of a high-risk exposure..

Digital proximity tracing **is a complement, not a substitute, for manual contact tracing**. It can augment the efforts of health officials, gaining precious time as alerts can be sent automatically and can inform otherwise unidentifiable contacts of a COVID-19 positive person.

Non-goals

Our system does not aim to achieve the following goals:

- **Track positive cases**: The system does neither attempt to provide a mechanism to track users who report a positive COVID-19 diagnosis through the app, nor to ensure that confirmed positive cases comply with medical orders. We assume that users who have received a positive test result act responsibly and take necessary precautions. Therefore, we do not attempt to detect contacts of confirmed positive cases *after* their diagnosis nor do we attempt to detect or prevent misbehavior. In our view, the gain in utility (potentially detecting irresponsible behavior of few individuals) does not justify the loss of privacy for the majority of users who adhere to guidelines to protect their fellow citizens. Moreover, our system does not provide location-tracking functionality and cannot determine when a user is "in public."

- **Detect hotspots or trajectories of positive cases**: The system does not attempt to identify locations frequented by confirmed positive cases, which might increase others' risk of exposure. This is a deliberate design decision. We limit the purpose of our system to its primary goal. This choice enables us to collect and process very little data. In particular it avoids collecting location data, which is highly sensitive and very difficult to publish in a privacy-preserving way.

- **Sharing data for research purposes**[6]: The system is not designed to support epidemiological research. As a side effect of fulfilling their primary purpose, proximity tracing systems produce data about close-range proximity between personal smartphones that might be of great value to epidemiologists and related research groups. This has triggered a public debate about whether proximity tracing systems should be designed specifically to collect additional data that might

[5] US CDC How COVID-19 spreads:
`https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/how-covid-spreads.html`

[6] It is in theory possible for proximity tracing systems to additionally share data intended for research purposes. However, this would invalidate the existing security and privacy analyses, and would require additional in-depth investigations into the impact of the shared data and the interaction with the other functions of the system.

help epidemiologists improve their understanding of and predictions about the spread of SARS-CoV-2.

We strongly believe, however, that it is not the time to conflate novel, untested technologies with the understandable desire to collect new epidemiological insights. Furthermore, the data collected by proximity tracing applications does **not** allow inferences about causal transmission chains (who infected whom), fomite transmission (transmission through surfaces of objects), or aerosol transmission (transmission via aerosols that remain suspended in the air for some time). We thus designed a system that is optimised to fulfill its primary purpose and to support and complement manual contact tracing through measurement of proximity over a prolonged period of time. How much of this data should be shared to support epidemiological research purposes is a separate question. We plan to publish a separate analysis of the privacy implications of data sharing.

A.3 System requirements

1) Enable proximity tracing To fulfill its primary purpose, the application must provide the following properties:

- **Completeness:** The exposure history captures all exposure events.
- **Precision:** Reported exposure events must reflect actual physical proximity.
- **Authenticity:** Exposure events are authentic, i.e., users cannot fake exposure events.
- **Confidentiality:** A malicious actor cannot access the contact history of a user.
- **Notification:** Individuals can be informed about prolonged exposure to the virus.

2) Respect and preserve digital right to privacy of individuals It is of paramount importance that any digital solution to proximity tracing **respects the privacy of individual users and communities** and **complies with relevant data protection guidelines** such as the European General Data Protection Regulation (EDPB Statement on GDPR and COVID-19) or the related Swiss law. The GDPR does not prevent the use of personal data for public health, particularly in times of crisis, but it still imposes a binding obligation to ensure that 'only personal data which are necessary for each specific purpose of the processing are processed' (art 25). It is therefore a legal requirement to consider, particularly in the creation of systems with major implications for rights and freedoms, whether such a system could be technically designed to use and retain less data while achieving the same effect. To this end, an application must minimize the amount of data collected and processed to avoid risks for individuals and communities, and it should reveal only the minimum information truly needed to each authorized entity.

Furthermore, a common concern with systems such as these is that the data and infrastructure might be used beyond its originally intended purpose. Data protection law supports the overarching principle of '**purpose limitation**' — precluding the widening of purposes after the crisis through technical limitations. Such assurances will likely be important to achieve the necessary level of adoption in each country and across Europe, by providing citizens with the confidence and trust that their personal data is protected and used appropriately and carefully. Only applications that do not violate a user's privacy **by design** are likely to be widely accepted.

The system should provide the following guarantees:

- **Data use:** Data collection and use should be limited to the purpose of the data collection: proximity tracing. This implies that the design should avoid collecting and using any data, for

example geolocation data, that is not directly related to the task of detecting a close contact between two people.

- **Controlled inference:** Inferences about individuals and communities, such as information about social interactions or medical diagnosis, should be controlled to avoid unintended information leakage. Each authorised entity should only be able to learn the information strictly necessary to fulfill its own requirements.
- **Protect identities:** The system should collect, store, and use anonymous or pseudonymous data that is not directly linkable to an individual's identity where possible.
- **Erasure:** The system should respect best practices in terms of data retention periods and delete any data that is not relevant.

3) Fulfill the scalability requirements posed by a global pandemic SARS-CoV-2 is rapidly spreading across the globe following people across national borders and continents. As a core principle of free democratic societies, after the current confinement measures end, free movement should resume. Proximity tracing must support free movement across borders and scale to the world's population.

The system should provide the following guarantees:

- **Scalability:** The system scales to billions of users.
- **Interoperability:** The system works across borders and health authorities.

4) Feasibility under current technical constraints There is an urgency to not only design but ***implement*** a digital system that simplifies and accelerates proximity tracing in the near future. This mandates a system design that is ***mindful of the technical constraints*** posed by currently available technologies.

- **No reliance on new breakthroughs:** The system should, as far as possible, only use techniques, infrastructure, and methods readily available at the time of development and avoid relying on new breakthroughs in areas such as cryptography, GPS localisation, Bluetooth or Ultra Wide Band distance measurements; or new deployments such as novel anonymous communications systems that have not been widely tested for privacy.
- **Widely available hardware**: The goal of high adoption of proximity tracing can only be achieved if both server- and client-side applications can run on widely available smartphones and server hardware.

B Decentralized proximity tracing

We propose a privacy-friendly, decentralized proximity tracing system that reveals minimal information to the backend server. We propose three different protocols to support exposure detection and tracing. These protocols provide developers with choice regarding the trade-off between privacy and computation cost but share a common framework.

In all three protocols, smartphones locally generate frequently-changing *ephemeral identifiers* (EphIDs) and broadcast them via Bluetooth Low Energy (BLE) beacons. Other smartphones observe these beacons and store them together with a time indication and measurements to estimate exposure (e.g., signal attenuations). See Figure AA.

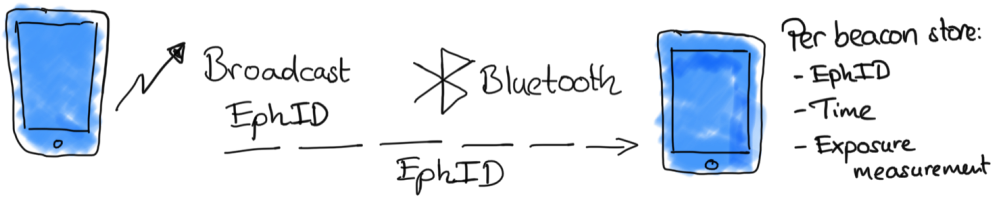


Figure 1: Processing and storing of observed beacons.

The proximity tracing process is supported by a *backend server* that distributes anonymous exposure information to the app running on each phone.[7] This backend server is trusted to not add information (i.e., to not add fake exposure events) nor remove information (i.e., to not remove exposure events) and to be available. The backend acts **solely** as a communication platform and does not perform any processing. It is **untrusted with regards to protecting users' privacy**. In other words, the privacy of the users in the system does not depend on the actions of this server. Even if the server is compromised or seized, their privacy remains intact.

If patients are diagnosed with COVID-19, they will be authorized by health authorities to publish a protocol-specific representation of their EphIDs for the contagious period to aid in decentralized proximity tracing. We are aware that each country, and in some cases each country's regions, will have existing processes and systems in place to manage mass testing, to communicate between testing facilities and laboratories, and to inform patients. In a separate document,[8] we discuss three proposals for secure mechanisms to validate upload requests from personal devices to the central backend and evaluate their usability trade-offs. Here, we leave the authorisation mechanism abstract. We further note that some implementations of the system might skip the authorisation step altogether and rely on self-reporting. However, we strongly advise implementing one of the proposed authorisation mechanisms to achieve stronger security guarantees.

When authorized, users can instruct their phones to upload a representation of the EphIDs to the backend. The backend stores the uploaded *representations.* To protect COVID-positive users from network observers, all phones equipped with the app generate dummy traffic to provide plausible deniability of real uploads.

Other smartphones periodically query the backend for information and reconstruct the corresponding EphIDs of COVID-19 positive users locally. If the smartphone has recorded beacons corresponding to any of the reported EphIDs, then the smartphone's user might have been exposed to the virus. The smartphone uses the exposure measurements of the matched beacons to estimate the exposure of the phone's owner, see Section 4.

B.1 Low-cost decentralized proximity tracing

In this section, we present a low-cost protocol that has good privacy properties and very small bandwidth requirements.

Setup *Initial seed generation.* Let t be the current day. Smartphones generate a random initial daily seed SK_t for the current day t. We assume days correspond to UTC days.

[7] We assume that the MAC address of the phone changes every time the ephemeral identifier (EphID) changes to prevent prolonged tracking of smartphones.

[8] "Secure Upload Authorisation for Digital Proximity Tracing", The DP-3T Project, https://github.com/DP-3T/documents/blob/master/DP3T%20-%20Upload%20Authorisation%20Analysis%20and%20Guidelines.pdf

Creating ephemeral IDs (EphIDs) *EphID Generation*. Each day, smartphones rotate their secret day seed SK_t by computing

$SK_t = H(SK_{t-1})$,

where H is a cryptographic hash function. The smartphone will use the seed SK_t during day t to generate EphIDs.

To avoid location tracking via broadcast identifiers, devices should frequently change the ephemeral identifier EphID that they broadcast to other devices. We refer to the duration for which a device broadcasts the same EphID as an **epoch**. The length of an epoch, in minutes, is a configurable system parameter that we denote as L.

At the beginning of each day t, smartphones locally generate a list of n = (24 * 60)/L new $EphID_i$s to broadcast during day t. Given the day seed SK_t, each device computes

$EphID_1 \,||\, ... \,||\, EphID_n = PRG(PRF(SK_t, \text{“broadcast key”}))$,

where PRF is a pseudo-random function (e.g., HMAC-SHA256), “broadcast key” is a fixed, public string, and PRG is a pseudorandom generator (e.g. AES in counter mode) producing n * 16 bytes, which we split into 16-byte chunks to obtain the n ephemeral Bluetooth identifiers EphID for the day.

Smartphones pick a random order in which to broadcast the EphIDs during the day. Each $EphID_i$ is broadcast for L minutes.

Local storage of observed EphIDs and seeds SK_t For each received beacon, phones store:

- The received ephemeral Bluetooth identifier EphID.
- The exposure measurement (e.g., signal attenuation).
- The day on which this beacon was received (e.g., “April 2”).

Note that an EphID could be received multiple times and will result in multiple entries in the database (Figure **??**). For efficient storage, we propose to group these entries by EphID, resulting in 36 bytes per EphID. Given a very conservative estimate of 140k different observations over the course of 14 days (i.e., if epochs are 15 minutes, these are 100 different people observed per epoch), this would require around 6.1 MB.

In addition, each device stores the seeds SK_t it generated during the past 14 days. This parameter (i.e., 14 days), which defines the maximum period for which any data (both observed and generated EphIDs)is stored on the device, is a system parameter and is determined by guidance from health authorities.

Decentralized proximity tracing Once the health authority has authorised the proximity tracing for a confirmed COVID-19 positive user (Figure 2, step 1), the user can instruct their phone to send to the backend the seed SK_t and the day t corresponding to the first day in which the user was considered to be contagious (Figure 2, step 2). The start date of the contagious window t can either be determined by the health authority or the user might be trusted to manually enter this day.[9] Epidemiologists estimate that for COVID-19 the contagious window starts 1 to 3 days before the onset of symptoms.

After reporting the seed SK_t and day t for the first day of the contagious window, the smartphone deletes SK_t. It then picks a completely new random seed and commences broadcasting EphIDs derived from this new seed. This ensures that after uploading their past seed, users do not become trackable. Recall that our system does not attempt to track users after reporting a diagnosis because we assume users act responsibly. The new seed will thus only be uploaded if necessary, i.e., if after a second positive diagnosis the user is considered contagious.

Given the seed SK_t, everyone can compute all ephemeral identifiers EphID broadcast by the COVID-19 positive user, starting from day t by repeating the process described in EphID generation above.

[9] “Secure Upload Authorisation for Digital Proximity Tracing”, The DP-3T Project, https://github.com/DP-3T/documents/blob/master/DP3T%20-%20Upload%20Authorisation%20Analysis%20and%20Guidelines.pdf

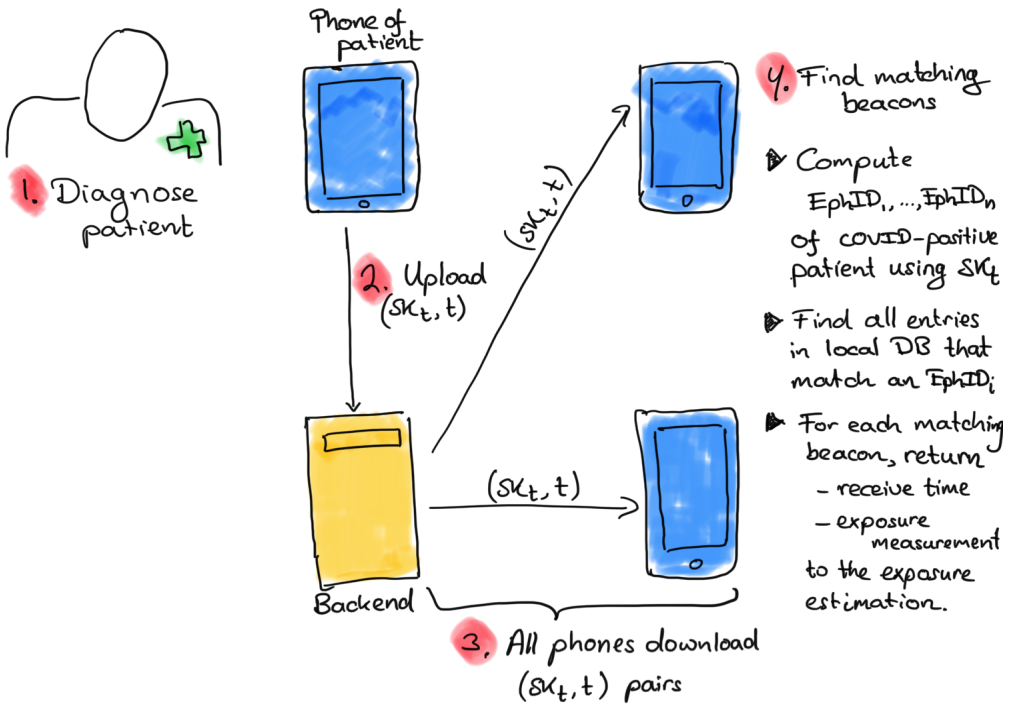


Figure 2: Proximity tracing process.

The backend collects the pairs (SK_t,t) of COVID-19 positive users. Phones periodically download these pairs (Figure 2, step 3). Each smartphone uses this pair to reconstruct the list of EphIDs of a diagnosed person for each day t' and checks (1) if it has observed any beacon with one of these EphIDs on day t' and (2) that such observations occurred before the corresponding seed SK_t was published.[10] Restricting the matching to a specific day limits replay attacks in which malicious users redistribute captured EphIDs and ensures more efficient lookups.

For each matching recorded beacon (e.g., a beacon with an EphID that was transmitted by a user who reported a positive diagnosis), the beacon's receive time and exposure measurement are taken into account for the exposure risk computation (Figure 2, step 4) and Section 4.

Scalability For each user who reports a positive diagnosis, the backend needs to store a 32-byte seed and a 2-byte day counter for the duration of the contagious window. Storage cost at the backend is therefore not a problem. Throughout the day, smartphones download the 32-byte seeds and 2-byte day counters of newly diagnosed patients. This data is static and can therefore be effectively served through a content delivery network (CDN).

B.2 Unlinkable decentralized proximity tracing

In this section, we present a variant of the low-cost design in the previous section that offers better privacy properties than the low-cost design at the cost of increased bandwidth. This design does not disseminate a

[10]To facilitate this check, the smartphone temporarily stores a more precise receive timestamp of all the beacons it received after the last download from the server. Once all downloads from the server have been processed, the phone coarsens this timestamp for all past observations.

list containing the seeds of users who have reported a positive diagnosis. Instead, the ephemeral identifiers of COVID-19 positive users are hashed and stored in a Cuckoo filter, which is then distributed to other users.

This design choice offers several advantages. It prevents adversaries from linking the ephemeral identifiers of COVID-19 positive users (see privacy analysis for details), and it enables COVID-19 positive users to redact, after a positive diagnosis, identifiers corresponding to sensitive locations, times, or periods in which users are certain they have not been in contact with other people, e.g. while they were alone or behind a window.

Setup No setup is needed.

Generating ephemeral IDs As in the low-cost design, smartphones broadcast each ephemeral identifier EphID during one epoch of fixed duration L. Epochs i are encoded relative to a fixed starting point shared by all entities in the system.

Smartphones generate the ephemeral identifier EphID_i for each epoch i as follows. The smartphone draws a random 32-byte per-epoch seed seed_i and sets:

$\text{EphID}_i = \text{LEFTMOST128}(H(\text{seed}_i))$,

where H is a cryptographic hash function, and LEFTMOST128 takes the leftmost 128 bits of the hash output. Smartphones store the seeds corresponding to all past epochs in the last 14 days. They delete older seeds. As before, the maximum storage period is a system parameter and is determined with guidance from health authorities.

Local storage of observed EphIDs For each observed beacon the smartphone stores:

- The hashed string H(EphID || i), where H is a cryptographic hash function, and EphID the identifier of the beacon, and i is the epoch in which the beacon is received. (Note that this differs from the low-cost design, in which the phone stores the raw EphID.)
- The exposure measurement (e.g., signal attenuation)
- The day in which this beacon was received (e.g., "April 2").

We include the epoch i in the hash to ensure that replaying an EphID outside the epoch in which it was originally broadcast can never cause a fake at-risk event (regardless of whether the EphID corresponds to a person who later reports a positive diagnosis).

For efficiency of storage, we propose to group these entries by hashed string. A single entry then requires around 52 bytes. Given a very conservative estimate of 140k different observations over the course of 14 days (i.e., 100 people observed per epoch), this would require around 6.9 MB of local storage.

Decentralized proximity tracing In case of a positive diagnosis, users can instruct their device to upload a representation of the EphIDs produced by the smartphone during the contagious window. Unlike in the low-cost design, the user first has the option to *redact* identifiers by choosing the set I of epochs for which they want to reveal their identifiers. For example, the user may want to exclude Monday morning and Friday night. The phone then uploads the set $\{(i, \text{seed}_i)\}$ for all epochs i in I. Requiring seed_i rather than the resulting EphID, ensures that malicious users cannot claim somebody else's EphID as their own (see security analysis).

Periodically (e.g., every 2 hours), the backend creates a new Cuckoo filter F and for each pair (i, seed_i) uploaded by a COVID-19 positive user it inserts

$H (\text{LEFTMOST128}(H(\text{seed}_i)) \,||\, i)$

into the Cuckoo filter F, i.e., the hashed string H(EphID || i) where $\text{EphID} = \text{LEFTMOST128}(H(\text{seed}_i))$ as above. The outer hash-function is needed for security. The backend publishes the filter. All smartphones download it.

Each smartphone uses the filter F to check if in the past (i.e., before the corresponding filter F was published), it has observed any of the EphIDs reported by COVID-19 positive users. The phone checks if any of its stored hashes are included in the filter F.

For each matching beacon (e.g., a beacon with a recorded hashed identifier that was transmitted by a user who reported a positive diagnosis), the beacon's receive time and exposure measurement are provided to the exposure risk computation (Section 4).

Cuckoo filters have a low, but non-zero, probability of false positives, that is reporting that they contain an element that was not in the input set. In order to avoid unnecessarily alarming users, we select the parameters of the Cuckoo filter such that false positives are highly unlikely to occur even with heavy usage of the system over a number of years. In the scalability calculation below, we configure the filter to produce one false positive in a million users over a period of 5 years.

The use of a Cuckoo filter hides the set of ephemeral identifiers of COVID-19 positive users from the general public. The system uses a Cuckoo filter in conjunction with inputs that are obtained from cryptographic hashes of random values (the seeds, concatenated with timestamps). The inputs to the filter are sparse in a large set, i.e., the set of all possible inputs (128-bit strings). These two factors makes enumeration attacks against the filter an unattractive attack vector for adversaries, while still making it possible for users who have observed particular ephemeral identifiers to check for their inclusion in the filter. Attacks that attempt to reverse the filter and directly recover inputs from values held in the filter do not result in exposure of ephemeral IDs because of the extra layer of hashing performed on ephemeral IDs before entering them into the filter.

Scalability This design requires more bandwidth and storage than in the low-cost design. The backend needs to store Cuckoo filters containing the hashed identifiers of COVID-19 positive users during the contagious period. Smartphones regularly download new cuckoo filters containing the latest hashed identifiers of COVID-positive patients. This data is static and can therefore be effectively served through a content delivery network. The computational cost on the phone is likely smaller than in the low-cost design, as phones only need to do one lookup per stored hashed identifier per cuckoo filter sent by the backend.

B.3 Hybrid decentralized proximity tracing

In this section, we present a hybrid design that combines ideas from the low-cost design and the unlinkable design. In this design, phones generate random seeds for each time window (for example, of length 2 hours) and use these seeds similar to the low-cost design to generate ephemeral identifiers for all epochs within that time window. Users upload seeds only if they are relevant to exposure estimation by other users.

Depending on the length of the time window, this design offers much better protection against linking ephemeral identifiers of COVID-19 positive users than the low-cost design and enables a user to redact time windows. The protection against tracking is weaker than the unlinkable design, but this scheme has a smaller bandwidth requirement.

This design is very similar to the Google/Apple design.[11] The Google/Apple design uses one seed to generate the ephemeral identifiers of that day, and thus corresponds to the special case where windows are 1 day long. In that configuration, the advantages with respect to the low-cost design are smaller. We recommend a time window of 2 or 4 hours depending on the bandwidth availability in the region.

Setup No setup is needed.

[11] See https://www.apple.com/covid19/contacttracing/

Generating ephemeral IDs As in the previous designs, smartphones broadcast EphID during an epoch of fixed duration L. We group consecutive epochs into a time window w. The length of a time window can range from 10 minutes to a full day and needs to be an integer multiple of L.

At the start of each time window w, smartphones pick a new random 16-byte seed seed_w. Given the seed seed_w for window w, each device computes

$\text{EphID}_{w,1} \,||\, ... \,||\, \text{EphID}_{w,n} = \text{PRG}(\, \text{PRF}(\text{seed}_w, \text{“DP3T-HYBRID”})\,)$,

where PRF is a pseudo-random function (e.g., HMAC-SHA256), “DP3T-HYBRID” is a fixed and public string, and PRG is a pseudorandom generator (e.g. AES in counter mode) producing n * 16 bytes, which we split into 16-byte chunks to obtain the n ephemeral Bluetooth identifiers EphID for the window w.

Smartphones pick a random order in which to broadcast the n ephemeral Bluetooth identifiers within the window w. Each EphID_i is broadcast for L minutes.

Local storage of observed EphIDs Smartphones locally record the observed beacons, similar to the low-cost design. For each received beacon the phone stores:

- The received ephemeral Bluetooth identifier EphID,
- The exposure measurement,
- The time window w in which the EphID was received.

For efficiency of storage, we propose to group these entries by EphID, resulting in 36 bytes per EphID. Given a very conservative estimate of observing 140k different EphIDs over the course of 14 days (i.e., if epochs are 15 minutes, this would be 100 people observed per epoch), this would require 4.8 MB of local storage.

Decentralized proximity tracing In case of a positive diagnosis, users can instruct their device to upload the relevant seeds seed_w generated by the smartphone during the contagious period. If the phone did not observe any EphID sufficiently close to be considered as an exposure during a time window w, it does not upload the corresponding seed seed_w for efficiency. As in the unlinkable design, the user additionally has the option to **redact** identifiers, by choosing the set W of windows for which they want to reveal their identifiers. For example, the user may want to exclude windows for Monday morning and Friday night. The phone then uploads the set $\{(w, \text{seed}_w)\}$ for all windows w in W.

The backend collects pairs (w, seed_w) of COVID-19 positive users. Phones periodically download these pairs. Each smartphone uses these pairs to reconstruct the list of EphIDs of COVID-19 positive users for each window w' and checks if it has observed any of these EphIDs during window w' *in the past* (i.e., before the corresponding seed seed_w was published). Restricting the matching to a specific time window limits replay attacks in which malicious users redistribute captured EphIDs and ensures more efficient lookups.

For each matching recorded beacon (e.g., a beacon with an EphID that was transmitted by a COVID-19 positive user), the beacon's receive time and exposure measurement are given as inputs to the exposure risk computation, see Section 4.

Scalability This design requires more bandwidth and storage than the low-cost design, but less than the unlinkable design. The backend needs to store the (w, seed_w) pairs corresponding to COVID-19 positive users. Smartphones regularly download all new pairs. This data is static and can therefore be effectively served through a content delivery network.

The download cost depends on the length of the window and how many windows can be automatically omitted by the smartphone when uploading seeds. See Figure **??** for a comparison. See the next section for how we computed these numbers.

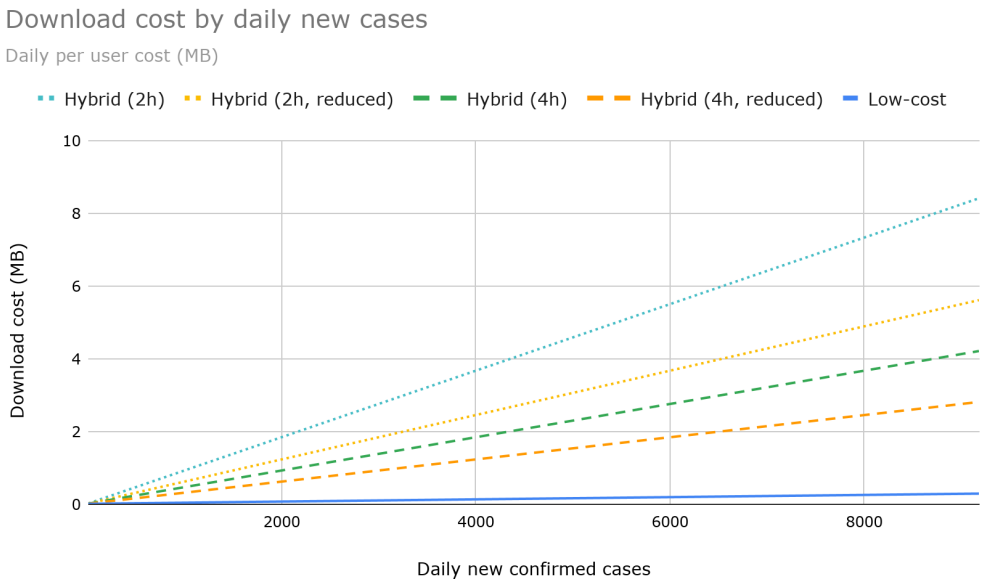


Figure 3: **Scalability of the hybrid design**. Comparison of the daily download cost per user (MB) depending on the number of new confirmed cases per day for different upload configurations of the hybrid design. We compare the download cost for different lengths of the time window w under two different assumptions. In the "normal" case, COVID-positive users upload seeds for all windows. In the "reduced" case their smartphone automatically omits the seeds for windows with a total length of 8 hours (e.g., because they were alone during that time and the phone did not detect any contacts).

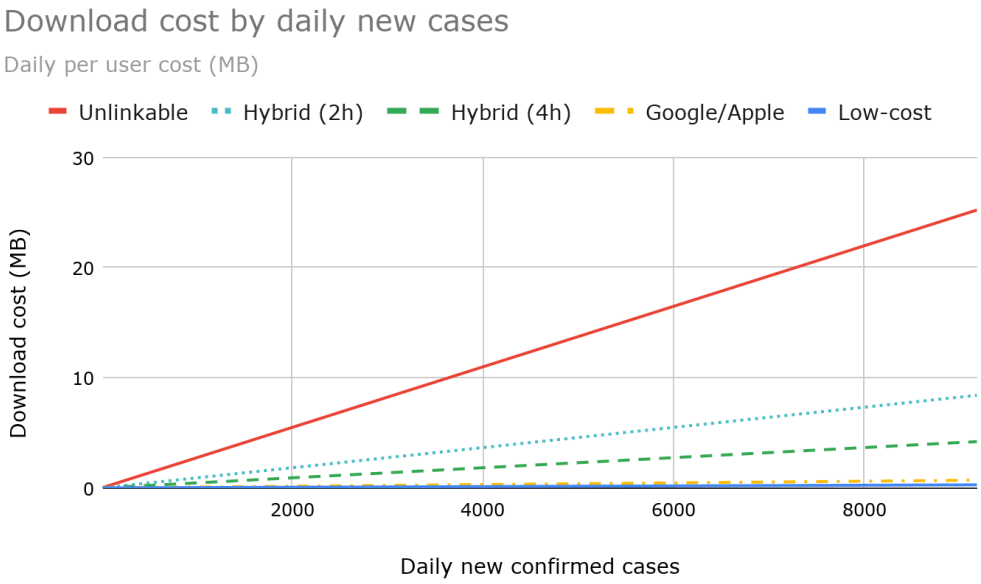


Figure 4: Comparison of the daily download cost per user (MB) by number of new confirmed cases per day for different decentralised proximity tracing designs.

B.4 Scalability

All three designs benefit from the use of a content delivery network (CDN). Smartphones of COVID-positive patients upload a small amount of data to the backend server. The backend server regularly redistributes this data to all other smartphones using a CDN. The daily download size scales linearly with the number of COVID-positive patients in all three designs. We assume a contagious window of 5 days and 15-minute epochs.

- In the low-cost design, the server needs to distribute one (SK_t, t) pair per diagnosed patient. This requires 36 bytes per patient.
- In the unlinkable design, the server needs to distribute 5 * 96 hashed strings per diagnosed patient. When using a well-tuned Cuckoo filter, this requires 2880 bytes per patient.
- In the hybrid design, we assume windows of 2 to 4 hours. The server must therefore distribute 5 * 6 or 5 * 12 seeds per diagnosed patient. This requires 480 to 960 bytes per user. Users that can redact 8 hours of windows only need to send 5 * 4 and 5 * 8 seeds, requiring 320 and 640 bytes respectively.

See Figure **??** for the resulting daily download cost for smartphones. Table DC shows specific values based on peak rates in several countries.

We expect that proximity tracing systems, even when deployed in large EU countries, will operate in the range of up to 2000 new cases a day. At the time of writing, all large EU countries see less than 1500 new cases a day. We expect that if the rate of new cases increases, countries will take policy measures to restrict the infection rates. Thus, we expect the download cost to never exceed the single-digit requirement.

	Low-cost (MB)	Unlinkable (MB)	Hybrid (4h) (MB)
Switzerland (8M)			
1390 cases	0.04	3.82	0.64
58 cases	0.00	0.16	0.03
Germany (83M)			
6294 cases	0.19	17.29	2.88
933 cases	0.03	2.56	0.43
France (67M)			
7578 cases	0.23	20.81	3.47
708 cases	0.02	1.94	0.32
Spain (42M)			
9181 cases	0.28	25.22	4.20
849 cases	0.03	2.33	0.39
Italy (60M)			
6557 cases	0.20	18.01	3.00
1402 cases	0.04	3.85	0.64

C Interoperability in decentralised proximity tracing systems

Effective proximity tracing systems must be interoperable across borders. Phones of users visiting foreign countries, whether it is for work, or for leisure, must be able to capture beacons from users in countries that they visit and include beacons of COVID-19 diagnosed patients in those countries in their exposure computation. Likewise, residents of a country must be able to receive notifications if a visitor to their country is diagnosed with COVID-19.

All three proposed designs support interoperability between different operators of different regions. Interoperability is possible as long as these operators use one of the decentralized designs proposed in this document. To interoperate with a specific protocol, the smartphone application must be able to process tracing data published for that protocol. That is, the application must run as many protocols as it needs to interoperate with.

To enable cross-border interoperability, backend servers of different regions (e.g., countries, or states) must exchange data. We propose the following mechanisms, explained in detail in other documents.[12]

First, when visiting a region, users enter the region into their phone. If a user visits a region frequently, e.g., workers commuting across borders, both regions can be permanently added to their phones. Phones use the list of visited regions to retrieve any proximity tracing data published by that region's backend for up to 14 days after the end of the users' visit. The way in which proximity data are published differs by protocol. For the low-cost design, this is a list of (SK_t,t) pairs, for the unlinkable design this is the cuckoo filter, and for the hybrid design this is the list of (w, $seed_w$)pairs. The phones then follow the protocol-specific procedures to match observed EphIDs, which they then feed into the exposure risk computation described in the next section.

Second, to ensure all contacts of diagnosed users are notified, when a user is diagnosed with COVID-19, the phone will ask the user for recently visited regions. When uploading the seeds to the server, the phones also supply the list of visited regions. The backend authenticates the upload and then redistributes the uploaded tracing data to all visited regions. As a result, users in those regions will download this data and can determine if they observed any of the visitors' EphIDs.

[12]For an overview see "Interoperability of decentralized proximity tracing systems across regions" retrieved from https://drive.google.com/file/d/1mGfE7rMKNmc51TG4ceE9PHEggN8rHOXk on 20 May 2020. A more detailed specification is provided here: "Decentralized Proximity Tracing: Interoperability Specification". The DP-3T Team (18 May 2020). Retrieved on 20 May 2020 from: https://github.com/DP-3T/documents/blob/master/DP3T%20-%20Interoperability%20Decentralized%20Proximity%20Tracing%20Specification%20(Preview).pdf

D Exposure estimation

The goal of the exposure estimation is to estimate the duration of the smartphone owner's exposure to COVID-19 positive users in the past. This measurement serves as a proxy for the level of exposure to the SARS-CoV-2 virus. Local health authorities determine the exposure threshold for when a user should receive a notification. A prolonged exposure to the virus does not imply that the virus has been transmitted. However, the notification serves as a trigger for precautionary interventions recommended by local health authorities, such as testing or quarantine.

To compute exposure, the smartphone proceeds as follows. First, if necessary, it downloads the latest parameters provided by the health authority. Next, it takes all matches reported by the proximity tracing system for the past 14 days and estimates the exposure. In Switzerland, for instance, for each day, the phone combines the exposure measurements (e.g. signal attenuations) of all matches corresponding to that day, to compute a per-day exposure score.[13]

If the exposure score is above the threshold determined by the health authority, the smartphone displays a notification that the user has been exposed to the virus through prolonged physical proximity to COVID-19 positive individuals. The notification advises the user on what to do and where to find more information. The details of the messages displayed, including the rate of repeated notifications and their content, need to be designed in close collaboration with health authorities and mental health experts.

E Security and privacy considerations

In this section, we analyse the privacy and security properties of the three decentralised proximity tracing protocols introduced in this document. We have published a separate, far more extensive, risk evaluation of digital proximity tracing systems[14] that includes the class of decentralised systems our three designs belong to.

E.1 Threat model

In this section, we describe the adversaries that we take into account when carrying out the security and privacy analysis. For each of these adversaries, we describe their capabilities and the kind of risk they pose for the system. In the next section, we analyze the security and privacy of the system with respect to these adversaries.

Regular user. A typical user of the system that is assumed to be able to install and use the application by navigating its user interface (UI). They exclusively look at information available via the app's UI to infer private information about other users.

Tech-savvy user (Blackhat/Whitehat hacker, NGOs, Academic researchers, etc.)**.** This user has access to the system via the mobile App. In addition, she can set up (BT, WiFi, and Mobile) antennas to eavesdrop locally. Finally, she can decompile/modify the app, and she has access to the backend source code.

- (Whitehat hacker) Will investigate the App code, the information in the phone, and will look at what information is exchanged with the server (using an antenna or software installed on the phone, e.g., Lumen) or broadcast via Bluetooth (passive).
- (Malicious) Can DOS the system (targeted or system-wide), deviate from protocols, and actively broadcast Bluetooth identifiers.

Eavesdropper (Internet Service Provider, Local System administrators, Bluetooth sniffer)**.** They can observe network communication (i.e., source and destination of packages, payload, time) and/or BLE broadcast messages.

[13] Details on the exposure estimation from BLE proximity measurements will be provided soon as separate documentation

[14] "Privacy and Security Risk Evaluation of Digital Proximity Tracing Systems", The DP-3T Project, https://github.com/DP-3T/documents/blob/master/Security%20analysis/Privacy%20and%20Security%20Attacks%20on%20Digital%20Proximity%20Tracing%20Systems.pdf

- (Network adversary) Can use observed network traffic to attempt to determine the state of a user (e.g., whether they are at-risk, COVID-19 positive, etc.).
- (Local Bluetooth BLE Sniffer) Can observe local Bluetooth broadcasts (possibly with a powerful antenna to cover a wider area) and try to trace people.

It should be noted however that in many instances, for individuals or companies to use data in this way, or to collect data about passers-by to try and estimate their infection status based on the announced identifiers, will fall foul of a range of existing national and European laws around data protection, ePrivacy and computer misuse.

Health authority. Receives information about COVID-19 positive users as part of their normal operations to diagnose patients. The health authority learns information about at-risk people only when these at-risk people themselves reach out to the health authority (e.g., after receiving a notification from their app).

Backend and App developers. Can access all data stored at the servers. Moreover, the backend can query data from the mobile app in the same way that it would do during normal operations (in our designs, it can only change the data downloaded by the smartphones). They could also change the code of their backend software and the code of the mobile apps (including parameters related to proximity tracing). We assume they will not modify the mobile app because such action would be detectable. They can combine and correlate information, request information from apps, combine with other public information to learn (co-)location information of individuals.

State-level adversary (Law enforcement, intelligence agencies, etc)**.** Has the combined capabilities of the tech-savvy user and the eavesdropper. In addition, a state-level adversary can obtain subpoenas that give them the capabilities of the health authority, or the backend. Their goal is to obtain information about the population or to target particular individuals. They may be interested in past information, already stored in the system, or future information that will enable them to trace target individuals based on observed EphIDs.

Unlimited-budget adversary. An adversary with an unlimited budget, e.g., large organizations and (foreign) nation states, has the capabilities of tech-savvy users but can deploy these at a much larger scale. Additionally, such an adversary might be able to gain control over the project's infrastructure such as the backend. The goals of this adversary might be to learn information about the population or individuals (cf. a state-level adversary) or to disrupt the proximity tracing system, resulting in a form of denial of service. One form of disruption is to try cause a sizable part of the population to receive fake at-risk notifications by deploying antennas in public locations (e.g., airports, train stations, shopping malls, or parliament buildings) and relaying messages far and wide. This relay attack increases the chances of "at risk" contacts for the targeted population because their phones will perceive proximity where there is none.

E.2 Privacy

E.2.1 Privacy concerns

Social graph. The social graph describes social relationships between users. Each node in the graph represents an individual user and an edge connecting two nodes indicates that there is a social relationship between the two users. A proximity tracing system does not need to provide information on the social graph to any party to fulfill its primary purpose.

Interaction graph. The interaction graph reflects close-range physical interactions between users. A labelled edge indicates an interaction between two adjacent users at a specific time. Knowledge of this graph is not necessary for proximity tracing nor for analyzing the spread of SARS-CoV-2. Therefore, *no party* needs to learn the interaction graph.

Location traceability. To perform proximity tracing, location traces (e.g. GPS coordinates) are not required. Therefore, no party in the system needs to have access to them or be able to easily trace individuals based on the BLE signals that the app broadcasts.

At-risk individuals. At-risk individuals are people who have recently been in contact with somebody who has tested positive for COVID-19. At-risk individuals need to know that they have been exposed to the virus so that they can take appropriate measures. No other party in the system needs to learn this information, other than when the notified user contacts and identifies herself to the health authority.

COVID-19 positive status. Only the user and the health authority need to know that the user has tested positive for COVID-19. No other party in the system needs to learn this information. In particular, app users do not need to know which of the individuals with whom they have been in contact have tested positive.

(Highly) Exposed locations. The system does not need to reveal any information about the locations that COVID-19 positive individuals have visited or the number of positive cases that have visited a specific location (e.g., to build a heat map of exposures). Proximity tracing can be performed without any party learning this information.

E.2.2 Privacy analysis of low-cost design

Social graph. The low-cost design does not reveal any information to any entity. Any two users involved in a contact may learn this contact's existence from the system, but this was already known to them.

Interaction graph. The system does not reveal any information about the interaction between two users to any entity. The EphIDs revealed by COVID-19 positive users do not allow any inference about the people they have been in contact with to anyone except those contacts. The system thus prevents outside parties from learning the interaction graph.

Location traceability. In our low-cost design, the EphIDs of all users are unlinkable, and only the smartphone that generated them knows the corresponding seeds SK_t. When the phone's owner is diagnosed with SARS-Cov-2 and gives permission, the phone publishes to the backend the seed SK_t corresponding to the first contagious day. After disclosing this information, the phone will generate a new seed at random. Given the seed SK_t of the first contagious day, the EphIDs of a COVID-19 positive user are linkable from the start of the contagious window until the time of upload (at which point the phone picks a new seed).

As a result, tech-savvy users, eavesdroppers, and state-level adversaries can *locally* track infected patients during the (past) window in which the identifiers broadcasted via Bluetooth are linkable. To do so, the attacker uses strategically placed Bluetooth receivers and recording devices to receive EphIDs. The app's Bluetooth broadcasts of non-diagnosed users and COVID-19 positive users outside the contagious window remain unlinkable.

At-risk individuals. The seeds revealed to the server by COVID-19 positive users *are independent of* their contacts, i.e., the people they interacted with. They therefore do not give any information about people at risk to any party other than the at-risk individuals themselves.

COVID-19 positive status. Any proximity tracing system that informs a user that she has been in contact with a confirmed positive case inherently reveals a piece of information to the person at risk: one of the people they interacted with has tested positive for COVID-19.

A curious or malicious adversary might attempt to exploit this and other information in the system to identify the COVID-positive individuals with whom they have been in close proximity.

A curious user who only uses the standard interface of the app, cannot learn which of their contacts has tested positive because the app in normal operation does not reveal any information other than that the user was exposed at some point in the past. Such a curious user can only learn which of their contacts has tested positive if they learn this fact on an out-of-band channel (e.g., the COVID-positive person informs them, they observe the person going to the hospital, a common friend reveals the COVID-positve status, etc.).

A proactive tech-savvy adversary can abuse any proximity tracing system to identify individuals who have reported a positive diagnosis to the system and that she has been in close proximity with. This risk is a consequence of the basic proximity tracing functionality. The attack can be executed regardless of implementation and proximity tracing protocol (BLE or otherwise). It only relies on the single bit of information that any proximity tracing system must reveal — whether you have been exposed to a confirmed COVID-19 positive case.

To reveal an individual's COVID-19 positive status, the adversary must (1) keep a detailed log of who they saw and when, (2) register many accounts in the proximity tracing system, and (3) use each account for proximity tracing during a short time window. When one of these accounts is notified, the attacker can link the account identifier back to the time-window in which the contact occurred to learn when she was in close proximity to an individual who reported a positive diagnosis**.** The attacker can correlate this information with their detailed interaction log to narrow down who in their list of contacts is COVID-19 positive. In some cases, the adversary might even be able to single out an individual. This attack **is inherent to any proximity-based notification system,** as the adversary only uses the fact that they are notified together with additional information gathered by their phone or through other means.[15]

In decentralized proximity tracing systems, such as the three designs we propose in this white-paper, tech-savvy adversaries can learn when they were in close proximity to a COVID-19 positive individual without having to create multiple accounts. To determine when they interacted with a COVID-19 positive individual, they **proactively modify the app**[16] to store detailed logs of which EphID they received and when, and cross reference this list with the EphIDs reported by COVID-19 positive cases downloaded from the backend server. They then correlate exposure times with their log of who they saw to reveal which individuals they have been in contact with reported a positive diagnosis.

The low-cost design allows an adversary to **link** the EphIDs reported by COVID-19 positive cases, i.e. to learn which Bluetooth identifiers belong to the same device**.** COVID-19 positive individuals upload a single seed SK_t that enables others to reconstruct, and thus link, a person's EphIDs for the entire contagious period. Due to the linkability of reported EphIDs, an attacker can **combine observations** at different times to identify who reported a positive diagnosis to the system. For example, the attacker might learn that the infected person she saw at 10:11AM is *the same* as the one she saw at 14:14PM. While she may have encountered many different people at each time, the intersection might be much smaller. This further increases the likelihood that the attacker can successfully single out a COVID-19 positive individual.

Tech-savvy users can also conduct a retroactive attack in which they attempt reidentification based on linkage and stored data, without the need to collect additional information in advance. The retroactive attacker only uses information stored by the app and auxiliary knowledge about the whereabouts of individuals during the contagious period. The data stored in the app provides coarse timing information when a specific EphID has been observed, e.g., per day in the low-cost design. A tech-savvy adversary could leverage this information to single out a COVID-19 positive individual based on matching observed EphIDs to background knowledge of whom the adversary was with during this time window. A combination of multiple time windows might be enough to uniquely identify to whom the reported EphIDs belong. However, since smartphones broadcast the daily set of EphIDs in random order, the attacker cannot use the published seeds SK_t to narrow down this coarse time-window. This decreases the likelihood that she will be able to successfully identify the COVID-19 positive individual in her contact list.

To **re-identify** an individual who has reported a positive diagnosis for COVID-19 to the system, an adversary needs to be able to **associate an identity** to the auxiliary information they have collected. For instance, a tech-savvy adversary who collects a detailed log of who they saw when needs to associate identities to each log entry. Without knowing the identities, the adversary cannot learn who tested positive. We can divide individuals the adversary interacts with into three groups depending on whether she will be able to reveal their identities or not:

- *Close individuals*: Family, friends, or colleagues with whom the adversary spends long periods of time. If these people received a positive diagnosis, they *will inform the adversary personally*

[15] For further details on this attack see "Privacy and Security Risk Evaluation of Digital Proximity Tracing Systems", The DP-3T Project, https://github.com/DP-3T/documents/blob/master/Security%20analysis/Privacy%20and%20Security%20Attacks%20on%20Digital%20Proximity%20Tra

[16] We note that in some schemes such modifications would preclude the App from accessing measurement data entirely when using the Google and Apple API.

about their diagnosis if they have spent time together. It is common practice that the authorities ask COVID-19 patients to notify any contact person at risk they remember.

- *Routine-sharing individuals*: People who share an activity with the adversary, such as riding a bus every day, supermarket tellers, etc. COVID-19 positive individuals in this group will likely not remember having been in contact with them and therefore will not (and cannot) notify the adversary.

- *Anonymous individuals:* People that the adversary sees sporadically and whose identities are unknown to the adversary.

As close individuals will reveal themselves, there is no extra information that an adversary can gain about the COVID-19 positive status of this group by exploiting the app. Anonymous COVID-19 positive users cannot be easily identified. Their privacy is only at risk if the adversary deploys additional (costly) means to associate identities with collected background knowledge. For instance, the adversary could attempt to combine data from surveillance cameras with facial recognition techniques to learn who is whom. The main group that is thus at risk through identification attacks is routine-sharing individuals.

We stress that in any case, having been close to an COVID-19 positive person is not proof of causality regarding transmission of the virus. Moreover, it is worth noting that reidentifying individuals and inferring their health status as a private entity without their permission would likely violate data protection law and, potentially, computer misuse law, which would further increase the cost and risk of undertaking this attack.

The pattern associated with the upload of identifiers to the server would reveal the COVID-19 positive status of users to network eavesdroppers (ISP or curious WiFi provider) and tech-savvy adversaries. If these adversaries can bind the observed IP address to a more stable identifier such as an ISP subscription number, then they can de-anonymize the confirmed positive cases. This can be mitigated by using dummy uploads. These dummy uploads provide plausible deniability to actual users' uploads, i.e., given an upload an observer cannot distinguish if it corresponds to an actual positive or a dummy. To avoid revealing which uploads were dummies to an adversary that polls the backend to learn if the list of ephemeral identifiers was updated, the backend should batch updates and only publish them in designated download slots.[17]

The backend server learns the IP address of COVID-19 positive users when they upload (a representation of) their EphIDs. If this adversary can bind the observed IP address to a more stable identifier, they can de-anonymize the confirmed positive patients. To reduce the risk, we recommend that the backend not log IPs.

*Mitigations***.** In the current setting, retroactive attackers can link beacons received at different, coarse times to aid in identifying COVID-19 positive users. The amount of information available to such an attacker can be reduced by running the proximity tracing protocol either inside a privileged OS-level module[18] or inside a local trusted execution environment (TEE). These approaches isolate the proximity tracing protocol and the data they collect from users and malicious apps. The protocols running in the isolated environment would only output for each matching beacon: the corresponding exposure measurement (e.g., the attenuation) and a coarse time. The app then computes the exposure score (see Section 4). As a result, retroactive attackers can only learn the number of beacons of infected patients received each day but can no longer link beacons by the same COVID-19 positive patient.

By itself, this approach does not protect against tech-savvy users that proactively modify their device to collect beacons and then compute matching COVID-19 positive beacons using the public list of COVID-19 positive EphIDs. However, when using TEEs to isolate the proximity protocol, the system can be extended to hide this public list from tech-savvy users, ensuring that they **cannot recognize** COVID-19 positive beacons. To protect against tech-savvy users when using TEEs, the backend encrypts the list of seeds so that this list can

[17]Details on the generation of dummy traffic will be provided in future documentation.

[18]This is the approach taken by the Google/Apple API.

only be decrypted *inside* the TEE. Each TEE downloads and decrypts the list of infected EphIDs and finds the matching beacons by cross referencing the list of infected EphIDs with the collected beacons. The TEE then returns to the app, for each day, a vector of the exposure measurements that enable the app to determine the user's exposure. As long as the TEE remains secure, tech-savvy users do not learn the EphIDs of COVID-19 positive patients.

Modern phones are equipped with TEEs that are used to harden smartphone kernels against attacks and to store cryptographic seeds. TEEs require buy-in from mobile platform providers (Apple, Google) and, for Android, the device manufacturers (Samsung, Huawei, etc.). The TEEs are well protected and difficult to attack even for tech-savvy users. While it is not impossible to leak this information, it is unlikely. We think such a mitigation is worthwhile in a later version of the proximity tracing system to further increase privacy guarantees. Other mitigation techniques could include the use of Private Information Retrieval and Private Set Intersection techniques, although current implementations may bring severe performance penalties.

Such technical measures as well as non-technical measures (e.g., banning modified applications from the market) could be introduced in case that the identification of COVID-19 positive individuals would become a threat to the system operation and to the users. The introduction of such measures depends on the overall risk assessment.

Finally, we note that if a small, cautious or misinformed portion of the population is concerned with these attacks and decides not to participate, this will not greatly impair the effectiveness of the deployment. As long as a large fraction of the population runs the app, the number of at-risk identifications will be large enough to significantly reduce the rate of transmission.

(Highly) Exposed locations. A powerful tech-savvy adversary operating its own BLE equipment from a single location can collect EphIDs within 20-100m range, depending on the phone output power and environment. When combining this list with the EphIDs that can be computed from the SKs downloaded to the phone, an adversary could learn whether any COVID-19 positive user has visited the location in a small radius of 50m. Furthermore, the adversary could reveal how many distinct diagnosed persons have visited the location in the past.

E.2.3 Privacy analysis of unlinkable design

The unlinkable design provides overall better privacy properties at the cost of increased bandwidth. The two designs provide the same level of protection for the social and interaction graph. We address the remaining differences point by point.

Location traceability. In the unlinkable design, the EphIDs remain unlinkable for all users against a local attacker. This unlinkability also holds for COVID-19 positive patients so long as the server is honest. However, if the server is malicious, then it can infer which ephemeral identifiers belong to a COVID-19 positive user through timing information or other metadata created when ephemeral identifiers are uploaded to the server. The use of anonymous communications could mitigate this threat.

At-risk individuals. As in the low-cost design, the seeds revealed to the server by users who have received a positive diagnosis *are independent of* their contacts. Hence, they do not give any information about people at risk.

The rest of the analysis is the same as for the low-cost design.

COVID-19 positive status. The unlinkable design reduces the linkability of the EphIDs reported by a COVID-19 positive user. Compared to the low-cost design, this reduces the likelihood that a proactive or retroactive tech-savvy adversary can identify which of their contacts has reported a positive diagnosis through linkage attacks. The adversary can no longer combine observations at multiple points of time to single out a COVID-19 positive individual.

Retroactive attackers can extract little information from the records stored on the phone. For each beacon, the phone only stores the hashed string, a coarse receive time (e.g., the day on which the beacon was received) and

the exposure measurement. Retroactive attackers cannot recover a more detailed receive time, and thus only learn the total count of COVID-19 positive beacons for each day.

However, a proactive tech-savvy adversary can still modify their application to learn when she has been in close proximity to a confirmed positive case. As described in our detailed privacy risk evaluation,[19] this attack can be executed in any proximity tracing system by using multiple accounts. It cannot be avoided.

The unlinkable design enables COVID-19 positive individuals to redact periods of time that they consider sensitive and for which they prefer not to disclose their contacts. This can alleviate concerns in a close-knit or small community in which users may be concerned that community members learn of their positive diagnosis through the app, instead of being informed in person.

(Highly) Exposed locations. As in any practical BLE-based PT system, an adversary could identify locations that have been visited by COVID-positive users in the past. However, as EphIDs cannot be linked to a single device, it is more difficult for the adversary to learn how many distinct cases visited the location.

E.2.4 Privacy analysis of hybrid design

The hybrid design provides privacy properties similar to the low-cost and the unlinkable design. There are two major differences between the hybrid and the low-cost design.

1) The hybrid design controls the linkability of the ephemeral identifiers reported by COVID-19 positive users and restricts linkability to short to medium-length time windows.

2) The hybrid design allows users who report a positive diagnosis to redact identifiers for specific time windows before sharing them with other devices.

These two differences affect the privacy properties of the hybrid design in the following ways:

COVID-19 positive status. Proactive and retroactive linkage attacks by tech-savvy adversaries aim to reveal the COVID-19 positive status of individuals they have been in close proximity with. To learn this information, the adversary needs to extract from the system *at which times* they have been in contact with a COVID-19 positive user. They can then correlate this information to auxiliary knowledge about who they saw when to identify individuals who reported a positive diagnosis. If the adversary can link ephemeral identifiers, i.e., associate multiple of the EphIDs reported by COVID-19 positive users to the same individual, the adversary can combine observations from different time frames to single out individuals. The hybrid design restricts the linkability of reported EphIDs to a time window w. This reduces the likelihood that the adversary can successfully narrow down the group of contacts who might have tested positive.

In comparison to the unlinkable design, the medium-term linkability of EphIDs implies the hybrid design provides slightly less protection against proactive and retroactive identification attacks. We note though, that as in all decentralized designs discussed in this white paper, the proactive tech-savvy user must modify their application to record receive times.

The hybrid design allows COVID-19 positive users to decide not to share their broadcast identifiers for certain time windows. This further reduces the likelihood of successful identification attacks as the adversary cannot use auxiliary information for the redacted time frames to reveal the identity of diagnosed users.

Retroactive attackers can extract some information from the records stored on the phone. For each beacon, the phone stores the EphID, a time and the exposure measurement. Because EphIDs from the same COVID-19 positive patient are linkable during a time window, the retroactive attacker can estimate contact duration with a single COVID-19 positive patient for each time window.[20]

[19] See "Privacy and Security Risk Evaluation of Digital Proximity Tracing Systems", The DP-3T Project, https://github.com/DP-3T/documents/blob/master/Security%20analysis/Privacy%20and%20Security%20Attacks%20on%20Digital%20Proximity%20Tracing%20Systems.pdf

[20] This retroactive attack does not work when using the Google/Apple API as it does not expose received EphID to any application or user.

(Highly) Exposed locations. An adversary is less likely to be able to learn the number of positive cases who visited a specific location because of the restricted linkability of ephemeral identifiers. The adversary can link EphIDs for the duration of a time window but cannot link identifiers of the same individual across multiple time windows.

E.3 Security

E.3.1 Security concerns

Fake exposure events. A fake exposure event could make a person believe that they are at risk, even though they have never been exposed to a diagnosed user. Attackers could try to generate fake exposure events to trigger false alerts, e.g. by relaying or broadcasting EphIDs at large scale. This would violate the authenticity requirement of the system.

Suppressing at-risk contacts: There is a risk that either a COVID-19 positive user or the backend server could prevent other individuals from learning they are at risk, e.g., by modifying the app's local storage. This violates the integrity of the system and would lead to an increased health risk for at-risk individuals who rely on the system for alerts.

Prevent contact discovery: A malicious actor could disrupt the system, e.g. by jamming Bluetooth signals, and prevent contact discovery.

E.3.2 Security analysis of low-cost design

Fake exposure events. In all practical proximity tracing systems based on Bluetooth-based exposure measurements, an adversary with a powerful antenna can trigger false alerts of an exposure to a COVID-19 positive person that do not reflect real-world proximity to a positive-case person.

To cause false alarms, a malicious adversary simply places her proximity tracing device in a crowded area and hooks up a powerful transmitter to *artificially increase the range* of her Bluetooth contacts. As a result, other devices located beyond 2 meters can interact with the attacker's device and will perceive the attacker's device as "near-by". To complete the attack, the attacker must ensure that these interactions between her device and other devices are flagged as exposure events. To do so, the attacker either:

1. **Herself tests positive** and brings her device to the hospital when she gets tested (requiring the adversary to be infected).
2. **Bribes a diagnosed person** to bring the attacker's device to the hospital instead of their own (or simply obtains the upload authorization code from them).
3. **Hijacks/bribes the health authority** that authorises COVID-19 positive individuals to trigger proximity tracing.
4. **Compromises the backend server** that sends information or directly notifies users of the system.

In the low-cost design, an attacker can record an individual's ephemeral identifier and broadcast it to victims at a different location and/or time, as long as it is **relayed on the same day**. If that individual later receives a positive diagnosis, the victims will incorrectly believe they have been exposed.

In the low-cost design, the seeds of COVID-19 positive users shared for exposure calculation are bound to the day on which they were valid. This prevents relay attacks in which the adversary attempts to relay an individual's ephemeral identifiers with a delay of more than 24 hours.

An attacker could be motivated to claim another user's EphID as their own and report that it should be included in the exposure risk calculation. The low-cost design addresses this risk by requiring users to upload the seeds SK_t from which their EphIDs are derived. As these EphIDs are derived from the seed using a cryptographic hash function and a pseudo-random function, it is computationally infeasible for an attack to learn another user's seed from observing their broadcasts.

Suppressing at-risk contacts. Hiding at-risk contacts is possible in any proximity tracing system. Infected users can choose to not participate at all; to temporarily not broadcast Bluetooth identifiers, or not to upload their data once diagnosed.

Prevent contact discovery. Any proximity tracing system based on Bluetooth low energy is susceptible to jamming attacks by active adversaries. Such jamming attacks will cause the normal recording of EphIDs to stop working, hence preventing contact discovery. This is an inherent problem of this approach.

E.3.3 Security analysis of unlinkable design

The unlinkable design has the same security properties as the low-cost design with respect to **suppressing at-risk contacts** and **preventing contact discovery**.

Fake exposure events. As in all practical proximity tracing systems based on BLE handshakes between personal smartphones, a powerful adversary can cause false alarms through BLE range extension attacks.[21]

In the unlinkable design, ephemeral identifiers are cryptographically linked to the *epoch* in which they are broadcast. To create fake exposure events, the attacker must therefore receive and rebroadcast EphIDs *within the same epoch.* Such an **"online" relay attack** is unavoidable in proximity tracing systems based on passive Bluetooth advertisements.

As in the low-cost design, the EphID generation protocol of the unlinkable design prevents an attacker from claiming another user's EphID as their own. To do so, an attacker would have to be able to infer a user's seed seedt from their broadcast identifier which is computationally infeasible.

E.3.4 Security analysis of hybrid design

The hybrid design has the same security properties as the low-cost design with respect to the risks of suppressing at-risk contact and preventing contact discovery.

Fake exposure events. As in all practical proximity tracing systems based on BLE handshakes between personal smartphones, a powerful adversary can cause false alarms through BLE range extension attacks.

In the hybrid design, EphIDs are linked to the valid time window of the seed they were derived from. To create fake exposure events, the adversary must therefore receive and broadcast EphIDs within the same time window. This prevents relay attacks in which the adversary attempts to relay an individual's ephemeral identifiers with a delay of more than the length of a time window.

As in the low-cost design, the EphID generation protocol of the hybrid design prevents an attacker from claiming another user's EphID as their own. To do so, an attacker would have to be able to infer a user's seed value $seed_w$ from their broadcast identifier which is computationally infeasible.

[21] For further details on this general attack see "Privacy and Security Risk Evaluation of Digital Proximity Tracing Systems", The DP-3T Project, https://github.com/DP-3T/documents/blob/master/Security%20analysis/Privacy%20and%20Security%20Attacks%20on%20Digital%20Proximity%20Tracing%20Systems.pdf

F Protection from short-term and remote eavesdropping at the physical layer

In this section, we introduce an enhancement to the decentralised proximity tracing solutions proposed in this document. It would also apply to similar initiatives such as PACT and TCN,[22] and the joint Apple and Google Exposure Notification protocol.[23]

A shortcoming in most decentralized proximity tracing systems based on the exchange of BT advertisements between devices is that a malicious party who is willing to modify their app or deploy their own software is able to record a proximity event ***despite only being in contact for a short amount of time or at a long distance***. This violates the requirement that the system provide ***precise*** data, i.e. only report exposure events that represent actual physical proximity.

In particular, an attacker could attempt to gather a significant number of EphIDs by deploying specialist equipment, either in high-traffic locations or in a vehicle that can cover a wide area ("wardriving"). The attacker can also deploy high gain, directional antennas to cover wide areas, further increasing the range and selectivity of the attack. The attacker can later see which of the recorded EphIDs correspond to users who reported a positive diagnosis and use additional metadata such as location, timing, video surveillance, etc. to infer their identities.

We note that for these enhancements to be efficient, changes in low-level smartphone components (e.g., Bluetooth chips) are likely necessary. We expect that without these changes, these enhancements will have a non-negligible impact on battery life.

F.1 EphID spreading with secret sharing

To address these problems, we introduce an enhancement to our system: EphID *Spreading With Secret Sharing*.

In a nutshell, this enhancement spreads each ephemeral identifier EphID across low-power beacons using a ***k-out-of-n secret sharing scheme***. Instead of transmitting each EphID within a single beacon, we encode it into n shares, such that each receiver needs to receive at least k shares to reconstruct the EphID. There are a number of secret sharing techniques that could be used for this purpose. We are currently running experiments to determine which is the most robust scheme for the scenarios in which these systems are to be deployed.

In our earlier designs, each device divides time into epochs and picks a random EphID_i for each epoch i. The EphID is transmitted several times during the epoch and each broadcast contains the whole EphID. The broadcast frequency depends on manufacturer- and implementation-specific details.

With this enhancement, each broadcast contains shares of the EphID. If EphID is to be retransmitted within an epoch, new shares are generated. In the simplest case, however, the number n of shares can be set to equal the number of broadcasts that the device makes within an epoch. We stress that this enhancement, even with the spread of the EphIDs, should not have a significant impact on battery life. It requires no additional transmission or reception over the basic designs, and the additional computation is minimal.

F.2 Tuning the trade-off between privacy and utility

Tuning privacy parameters. Setting the value k requires careful consideration. A system must receive enough beacons during the contact interval (as determined by epidemiologists) to receive k shares and register a contact. A smaller k thus increases the robustness of the system in normal operation. However, the larger the value of k, the longer an adversary is forced to shadow a victim in order to collect a sufficient number of beacons to reconstruct the EphID. Hence, the proposed number k of shares is a trade-off between privacy and the ability to record short contacts.

[22] For PACT, see Justin Chan et al. (2020) PACT: Privacy Sensitive Protocols and Mechanisms for Mobile Contact Tracing, retrieved from: https://covidsafe.cs.washington.edu/on 20 May 2020. For TCN, see TCN Coalition (2020) TCN Protocol, retrieved from https://github.com/TCNCoalition/TCN on 20 May 2020.

[23] See https://www.apple.com/covid19/contacttracing/

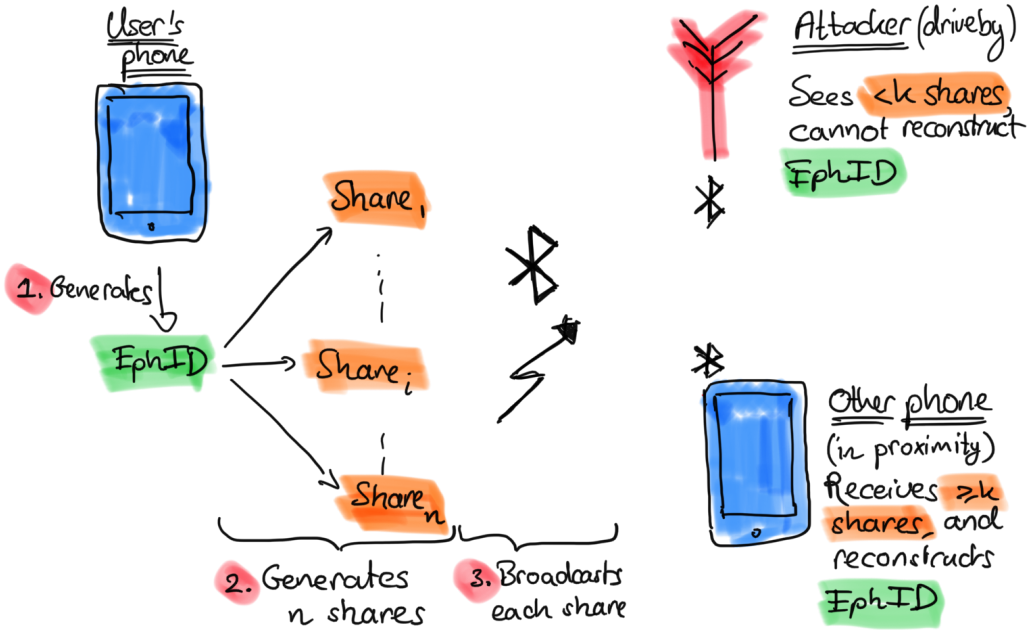


Figure 5: EphID spreading with secret sharing.

Eavesdropping from a distance. Spreading EphID across beacons not only protects against eavesdropping by adversaries who are only briefly collocated with their victim, but it also makes eavesdropping from a distance much harder. The requirement to successfully receive multiple broadcasts increases the asymmetry between a legitimate receiver in proximity and a malicious eavesdropper at a distance. An eavesdropper who is placed further away will typically experience a worse channel to the transmitter and a higher packet loss.

If we assume an attacker without access to specialised equipment and with reasonable assumptions on the broadcast transmission power, frequency, and probability of successful reception, we can select k, n such that a close by, legitimate user within 5 meters would have a very high probability of successfully receiving an EphID within a reasonable contact time threshold of five minutes ($>$99.9%), but an attacker attempting to eavesdrop from 16 meters away would have a small probability of success ($<$1%). An attacker using specialised hardware would be able to improve their odds either by increasing their probability of successful reception or by cryptographic analysis of the malformed broadcasts.

To achieve the appropriate balance between the desired range of reception of EphIDs (epidemiologically relevant) and the resilience to eavesdropping, we need to select the right combination of transmission power, transmission frequency, and required number k of reconstruction shares. We expect these parameters to be configurable and determined by further experiments, functional requirements, and risk assessment. The use of ultra-low or low power beacons will likely best protect the privacy of the users and facilitate proximity detection.

Our scheme can be integrated within a ranging technique or used in addition to existing (e.g., RSSI-based) ranging. In the latter case, ranging would use a different epoch identifier that is different but linked to the EphID that the device is broadcasting. If supported by BLE chipsets, this scheme could be further enhanced by in addition distributing the shares across three BLE advertisement channels.

G Comparison with centralized approaches

We classify two key functionalities in proximity tracing systems that are decentralized in our schemes:

- Ephemeral identifier generation: Ephemeral identifiers broadcast via Bluetooth are generated on the phone.
- Exposure estimation: The estimation of the exposure is computed locally on the phone.

We now compare the security and privacy properties of the decentralized approaches presented above with schemes in which both operations are centralized, and with schemes in which seeds are generated on phones but COVID-exposure estimation is centralized.

G.1 Centralized identifier generation and exposure estimation

In approaches in which both identifier generation and COVID-exposure estimation are centralized, such as ROBERT,[24] PEPP-PT-NTK,[25] and OpenTrace/BlueTrace/TraceTogether,[26] a central server estimates a user's likelihood of COVID-exposure, instead of the user's smartphone in decentralized designs. Depending on the system, the server notifies the at-risk users (PEPP-PT-NTK, OpenTrace) or users query the server about their status (ROBERT).

In all these systems, the central server holds a long-term pseudo-identifier for every user and uses it to derive ephemeral pseudo-identities (EphIDs) that are pushed to the smartphones.

The smartphones broadcast the EphIDs received from the central server and record the EphIDs transmitted by near-by smartphones. Smartphones *locally store all* observed EphIDs together with their corresponding proximity and duration. See Figure ZZ.

In case of a positive diagnosis, users can give permission for their smartphone to send the recorded list of observations to the server to enable proximity tracing.

G.1.1 Central proximity tracing

PEPP-PT-NTK and OpenTrace The PEPP-PT-NTK and OpenTrace backends execute the proximity tracing process after a diagnosed user has uploaded their list of observations [(EphID, epoch, duration)] for the contagious window. The backend recovers the long-term pseudo-identifiers of the at-risk users from the reported observed EphIDs and triggers a process to notify them if their exposure is high enough. See Figure ZY.

ROBERT In the ROBERT system, the backend tracks the exposure of each user of the system. As in PEPP-PT-NTK and OpenTrace, COVID-19 diagnosed patients upload their observed EphIDs to the backend server. The backend server associates these observed EphIDs to long-term pseudo-identifiers of at-risk users. The backend uses the associated data to update the exposure for each of the at-risk users.

Unlike PEPP-PT-NTK and OpenTrace, the backend does not notify patients. Instead, smartphones of ROBERT users regularly query the backend to request their exposure status. The backend answers whether the exposure passed the threshold or not.

[24] Inria PRIVATICS and Fraunhofer AISEC (19 April 2020) *ROBERT: ROBust and privacy presERving proximity Tracing*. Retrieved from https://github.com/ROBERT-proximity-tracing/ on 19 May 2020.

[25] PEPP-PT (20 April 2020) *Data Protection and Information Security Architecture: Illustrated on German Implementation*. Retrieved from https://github.com/pepp-pt/ on 19 May 2020.

[26] Jason Bay, Joel Kek, Alvin Tan, Chai Sheng Hau, Lai Yongquan, Janice Tan, Tang Anh Quy (9 April 2020) *BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders*. Retrieved from https://bluetrace.io/ on 19 May 2020.

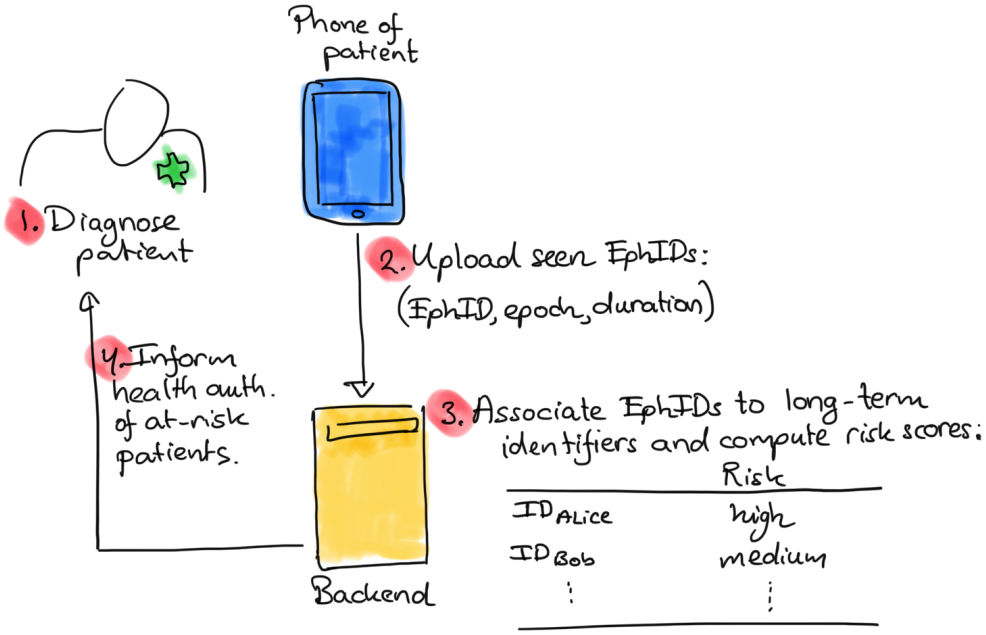


Figure 6: Processing and storing of observed EphIDs.

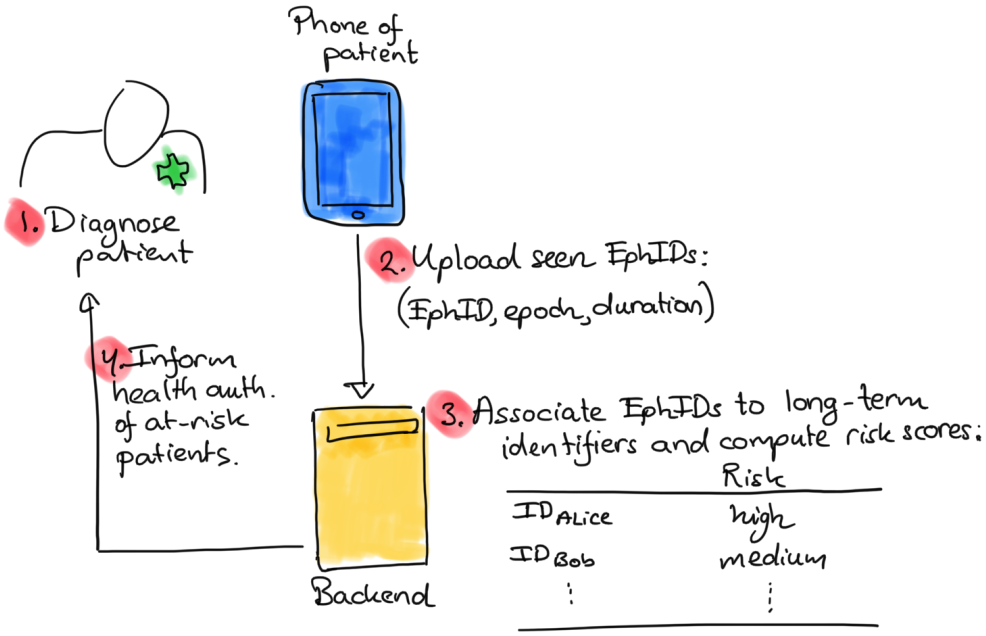


Figure 7: Proximity tracing for centralized PEPP-PT-NTK and OpenTrace designs. In ROBERT, the users instead query the backend.

G.2 Local identifier generation and centralized exposure estimation

Other approaches, such as DESIRE[27] instead generate identifiers on the phone, while still estimating COVID-exposure centrally. Instead of broadcasting self-contained, ephemeral identifiers, smartphones in DESIRE broadcast ephemeral public keys that, when combined with others' public keys, yield ephemeral identifiers EphIDs.

In case of a positive diagnosis, users can give permission for their smartphone to send a version of the observed EphIDs to the backend to enable proximity tracing.

Central proximity tracing Smartphones regularly query the backend to request their current exposure status. To enable the backend to compute this status, phones upload a version of the EphIDs computed from all the encounters they had in the relevant period. The server takes all observations reported of these encounters [(EphID, epoch, duration)] and estimates exposure. If the exposure is long enough, the user receives a positive exposure response. Otherwise, the user receives a negative.

G.3 Privacy comparison

Social graph. In a system in which both identifiers and COVID-exposure are computed centrally, the backend server can always associate ephemeral broadcast identifiers with permanent pseudo-identifiers for individual devices. If EphIDs are associated with a long-term identifier (e.g., in PEPP-PT-NTK), the backend server can reconstruct the social graph of users from the information shared by COVID-19 positive users. The server can join subgraphs from different positive cases to gain a comprehensive picture of the true underlying social graph. Given other partial social graphs with identities, the server can match its graph to the other graphs and reidentify nodes.

ROBERT and DESIRE propose to prevent the leakage of the social graph by using an anonymous communication network to upload observed identifiers in an unlinkable manner. In this way, the backend cannot associate uploads to a user nor determine which identifiers were observed by the same COVID-19 positive user. As a result, the backend cannot reconstruct an infected user's contacts.

In separate documents, we show that (1) these mechanisms when applied to ROBERT are ineffective and still allow reconstruction of the social graph;[28] and (2) are difficult to realise in practice for DESIRE.[29]

Interaction graph. In a system in which both identifiers and COVID-exposure are computed centrally and uploaded observed identifiers are linked, the backend server can always associate uploaded ephemeral broadcast identifiers to permanent pseudo-identifiers for individual devices. In PEPP-PT-NTK, OpenTrace, and ROBERT, observed identifiers are timestamped. Thus the backend server can not only reconstruct a social graph, but it can reconstruct an interaction graph.

The subset of the full interaction graph learned by a server grows quickly as every newly confirmed positive user uploads their entire contact history, which can be linked to existing nodes in the graph. Even though the nodes in the graph are pseudonymous, this is a serious privacy concern because graph data is easy to reidentify.[28]

If deployed (including anonymous communication, ensuring enough mixing with uploads of other COVID-positive patients, and anonymous authentication), the mechanisms in DESIRE to protect the social graph preclude the DESIRE backend from learning the interaction graph.

[27] Castellucia et al. (9 May 2020) DESIRE: A third way for a European Exposure Notification System https://github.com/3rd-ways-for-EU-exposure-notification/project-DESIRE/blob/master/DESIRE-specification-EN-v1_0.pdf on 19 May 2020.

[28] See, "Security and privacy analysis of the document 'ROBERT: ROBust and privacy-presERving proximity Tracing' ", The DP-3T Project, version 22 April 2020, https://github.com/DP-3T/documents/blob/master/Security%20analysis/ROBERT%20-%20Security%20and%20privacy%20analysis.pdf

[29] See, "DESIRE: A Practical Assessment", The DP3T Consortium, version 13 May 2020, https://github.com/DP-3T/documents/blob/master/Security%20analysis/DESIRE%20-%20A%20Practical%20Assessment.pdf

Location traceability. The decentralized design limits the potential for location tracking to users who have received a positive diagnosis and for the course of the contagious period. In centralised systems in which keys are generated on the server, access to server-side keys (e.g., the backend itself or law enforcement) enables linking ephemeral EphIDs to the corresponding permanent app identifier. This enables tracing/identifying people based on EphIDs observed in the past, as well as tracing peoples' future movements.

When keys are generated on the phone and not used directly as EphIDs, as in DESIRE, location traceability is equivalent to the decentralised unlinkable design.

At-risk individuals. In centralised systems in which the server controls key generation and notifies the user (PEPP-PT-NTK, OpenTrace), by design, the backend recovers the identity of an at-risk individual to notify these individuals.

In other centralised designs (ROBERT, DESIRE), users query the server to learn their exposure status. The identity of at-risk users is only protected when servers cannot deanonymize users through their permanent app identifiers and network identifiers.

As in decentralized designs, network eavesdroppers do not learn at-risk status.

COVID-19 positive status. The centralised and decentralised proximity tracing systems share the inherent privacy limitation that they can be exploited by a tech-savvy user to reveal which individuals in their contact list might be infected. However, the centralised designs hide when and how often the user was in contact with a COVID-positive patient. As a result, tech-savvy attackers cannot benefit from linking between EphIDs and timing information to amplify their attack. Instead, they need to rely on multiple accounts.

Depending on whether the centralized designs deploy dummy traffic correctly, network eavesdroppers might still learn the COVID-19 status of users of the system.

G.4 Security comparison

Fake exposure events. Triggering false alerts is easy in all centralised designs except DESIRE and can be done retroactively by any tech-savvy COVID-19 positive user. It does not require broadcasting. It suffices to add the target's EphIDs to the list of observed events prior to uploading them to the backend.

DESIRE requires an active exchange between users to trigger a fake exposure event, and therefore requires broadcasting.[30]

Suppressing at-risk contacts. Hiding at-risk contacts is possible in any proximity tracing system.

Prevent contact discovery. Any proximity tracing system based on Bluetooth BLE is susceptible to jamming attacks by active adversaries.

H Conclusion

In this whitepaper, we designed a privacy-preserving proximity tracing system and analyzed three different protocols. All three protocols minimize exposure of private data, limiting the risk of a privacy leakage.

Our design relies on smartphones to *locally* compute the exposure of an individual user to the SARS-CoV-2 virus through proximity over a prolonged period of time to COVID-19 positive people. Data about specific exposure events, i.e., interactions of people, always remains on a user's phone.

The three implementations offer different trade-offs between bandwidth and privacy protection. One design results in an extremely lightweight system. The others offer extra privacy properties at the cost of a small increase in download data size. The three alternatives scale to a large number of users with minimal local computation and minimal centralization.

[30] For further details on this general attack see "Privacy and Security Risk Evaluation of Digital Proximity Tracing Systems", The DP-3T Project, https://github.com/DP-3T/documents/blob/master/Security%20analysis/Privacy%20and%20Security%20Attacks%20on%20Digital%20Proximity%20Tracing%20Systems.pdf

We also provided evaluation criteria to assess the level of privacy provided by any proximity tracing solution. We thoroughly evaluated our protocols with respect to performance, security, and privacy. Compared to central designs in which the backend computes risks and informs users, our design protects interaction graphs from the backend. Only a determined, tech-savvy adversary can learn any extra information besides that made visible by the app. The centralized system, in comparison, leaks to the backend unnecessary information about contacts and requires a large amount of trust in a central entity.

Our three implementations show that there are a wide range of alternatives to be explored among the trade-off between resistance to different active and passive attacks, battery consumption, and bandwidth overhead. We encourage researchers and technology companies working on proximity tracing to continue searching for the best realistic operation point.

Contact Tracing: Holistic Solution Beyond Bluetooth

Ramesh Raskar, Deepti Pahwa, Robson Beaudry
raskar@media.mit.edu
MIT Media Labs

Abstract

Contact tracing is a critical part of reopening society. While manual contact tracing by medical professionals is essential, there's growing acknowledgement that supplementing this with digital approaches might make a significant difference in the speed with which we can reopen.

Safe Paths is open source, standards-based, privacy-first framework that works closely with public health entities. Our approach is to roll out apps, SDKs, privacy-preserving network backbones, and interoperable protocols, so that any developer can build experiences that perform contact-tracing and related activities in safe, easy to use ways.

In this paper, we'll compare some of the existing technologies that can be used to aid contact tracing and exposure notification efforts, and make the case for using a holistic, multi-modal approach rather than relying exclusively on a single technology.

A Bluetooth Tracing

The technology that's being most widely explored for exposure notification today is Bluetooth Low Energy transmission (BLE); it's present on most modern mobile devices, and can provide evidence that two phones were near each other. Several groups, including PACT, TCN, BlueTrace, D3PT, and others, have been working together in recent weeks to create privacy-preserving protocols that use Bluetooth for COVID exposure notification. Most recently, Google and Apple have collaborated on a framework for contact tracing apps, which will be extremely helpful in overcoming major barriers to adoption (including interoperability between the two companies' phones, improvements in battery life, and allowing the use of BLE in the background). (For the rest of this essay, we'll refer to this Google / Apple Exposure Notification protocol as "GAEN".) By building contact-tracing support into their operating systems, GEAN could make a difference in adoption of these technologies.

At the same time, there are a host of reasons why the GAEN protocol is not, on its own, a complete solution for contact tracing. Successful contact tracing requires a strong understanding of the context of encounters: where and when did an encounter happen, and what are the chances that the encounter resulted in a transmission? Bluetooth technology does a good job of detecting physical proximity, but it does not provide any other context of the encounters.

Furthermore, contact tracing (with or without proximity detection) is itself just one piece of the puzzle; public health officials need to be able to generate heatmaps, spread analysis, and other data that allow them to fight this

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

disease holistically. With an understanding of location and context, we can develop that more holistic solution. We need to carefully consider the role of various technologies, their context to the end-user, as well as to the health officials and communities, which we will explore throughout this document.

User experience and App Perspective: The need for a holistic solution with GPS, WiFi and Bluetooth There are several reasons to consider a holistic multimodal solution that includes location tracking in addition to Bluetooth proximity ID. Any one technology has limitations of either false positives, or false negatives. Ultimately, the best solutions will leverage the advantages of GPS, Bluetooth, and WiFi in order to create more accurate, useful, and private data.

Adoption Rate vs Effectiveness: Bluetooth requires many people to use the apps for it to be valuable. In Singapore the Bluetooth App penetration is 12% which means only 1.44% of encounters are recorded (0.12*0.12). As a reference, Casey Newton [1] has a good piece on Why Bluetooth apps are bad at discovering new cases of COVID-19. While adoption can be increased through government and big tech buy-in, these numbers still hamper effectiveness.

GPS based App scales linearly: With a 12% adoption rate of the app, considering a proportional approach, we might get much better than 12% coverage in any given area. This means that if 12% of people install the app. (I.e. if 100 infected people went to the store in a week, and we only caught 12 of them, that's still potentially enough to label it as a hotspot.12% of infected hotspots will be identified This identification would be very helpful, as nearly everyone in town will hear about them, eg. local news channels. GPS is useful even if smartphone penetration in a region is not widespread.

False positives or False Negatives: Attempting to do exposure alerting as the only intervention in such cases has the risk of generating either false positives, or false negatives. For instance, if you are living in the apartment below an infected person, sharing a ceiling/floor, or working from a neighboring office, such a notification can still flag you as at risk. False negatives could occur if your phone was turned off or not with you when you came into contact with an infected person. This ambiguity can only be handled by a good UX.

Contextual information is key: Users do not like to get an alert without understanding the context. They need to know where exactly the encounter took place in order to trust the system. This location information can be provided by GPS location logging. Note that adding location and time to BLE encounters make them less private, because in some cases it will allow the person receiving the alert to deduce from whom the alert was sent (if they know they were only in the presence of one other person at the time, for example). But without this contextual information, it's impossible for the exposed individual to judge the reliability of the notification.

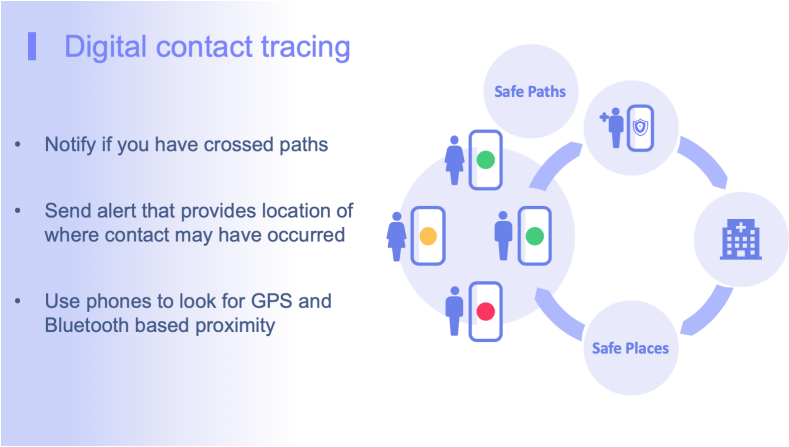
Further, in the context of manual contact tracing by medical professionals, access to location information can help them conduct interviews by helping the patients, for example, remember if they were wearing a mask or not, whether they shook hands or not, etc.

B Health Organisations Perspective: Public health needs more than just encounters

The prime need of the health officials is their ability to *call the exposed person directly to assess their personal situation* and provide guidance about testing/isolation/hospitalization. Health care officials also need *dashboards of emerging hotspots* to be able to map the spread. Safe Paths is working towards inclusion of building tools for heatmaps and spread analysis to support their needs. Bluetooth does have certain advantages, especially in dense

urban environments, but the optimal contact tracing solution will ultimately be multimodal (Bluetooth, GPS, and Wi-Fi), leveraging the advantages of each technology. For example, contextual information from GPS location logs can help a simple rejection mechanism for the false positives created by Bluetooth. As an example if there was a possibility to judge altitude and be confident that people are on different floors of a building, or if you could use velocity data to determine that someone is on a vehicle, perhaps you could reject an encounter. However, in case of velocity data, it would still be a challenge to know if it's not someone who got on the same bus as you for maybe one or two stops. GPS also allows for the creation of crucial virus heatmaps for health professionals, without needing large scale adoption by the population. Safe Paths is developing privacy preserving *self-reporting that will not lead to misinformation and abuse*. And self-reporting is very important as going to testing sites still has a lot of friction, especially among young people. While a Bluetooth API certainly makes aspects of contact tracing easier, public health officials need more than simply contact tracing to address the epidemic. They need a larger ecosystem that can help them with *health verification, patient interview*, hotspot identifying, and more. All of these tools furthermore need to protect privacy and be adapted for local conditions.

C Safe Paths is building Ecosystems: Solutions require more than just Contact Tracing



Contact tracing is just a tiny part of the public health interventions we need to build: for encounter memories, checking on loved ones or checking on co-workers you meet daily etc., health verification, sick leave certification and so on especially for restarting the economy. The tracing solution apps should be able to amplify the role of Public health officials.

SafePaths is, above all, about building open standard end-to-end solutions for citizens and public health while maintaining privacy and scalability. We will continue to use the best tools, Bluetooth APIs included, to achieve this aim. In the early phases, the emphasis is on rapid iteration and deployment for solutions for epidemic tracking. In the later phases, the goal is building encrypted computational methods that can be useful in any future societal disruptions. In [2] Prof. Raskar's 2019 talk about creating an honest impartial broker to address challenges in a fragmented society.

We are already developing pilots in 30 jurisdictions across the world. SafePaths is working closely with public health entities, and we plan to roll out apps and end-to-end solutions for each entity that makes use of various technologies including the recent Apple/Google Bluetooth API. We are furthermore building interoperable protocols and standards, so that each of these jurisdictions can benefit and learn from each other.

D Safe Paths is Technology Agnostic

We want to stress that Apple and Google are releasing Proximity AP but not a full contact tracing solution. The released APIs will give contact tracing tools such as Safe Paths the ability to work at a deeper level on phones, improving battery life, effectiveness and privacy.

Here, it might be important to bring to light that while there are many contact tracing solutions that could be built on these APIs, there are issues of concern with relying on either one of the technologies alone - Bluetooth or GPS based solutions. In the case of Bluetooth, each phone (not just the infected person's phone) is emitting the Bluetooth ID every few seconds. So if an app is not transparent or open source, or runs afoul of other important privacy principles, there's a great capacity for abuse, by listening in on the data of hundreds of millions of users as they broadcast IDs that change very slowly (15 mins). Smartphones can help reduce the spread of the virus but any network analysis could cause unacceptable intrusion to privacy and human rights.

Conversely, the underlying data set for location data is much riskier than for Bluetooth data; that is, if you did share location data without privacy protection (either because of a poorly designed app, or by insufficient security protection), the results would be much more damaging than the release of BLE data (which is composed only of anonymous encrypted IDs).

In this vein, MIT SafePaths is in the process of building out both private tracing and a public health solution, part of which will utilize the new Google/Apple API, defined by two key elements:

1. open source software components

2. interoperable standards and backbone that work across GPS/BT/WiFi/Telecom

We will be using either *on-device* calculation or use *encrypted trail match* with guarantees of privacy. Thus it avoids the 'big brother' surveillance state problem in certain countries where GPS matching solutions have been very effective but are draconian [3]. Such misinformation and distrust can cause civil unrest, especially in heterogeneous societies.

The vision at Safe Paths is to enable a fusion of various technologies including GPS/Bluetooth/WiFi SSID to provide a more reliable approach, reduce false positives/negatives, provide context for believability and also provide aggregate dashboard view for public health officials. SafePaths is already delivering these technologies and will continue to build them in an open source way. Follow CovidSafePaths.org for a more technical discussion.

References

[1] C. Newton, "Why bluetooth apps are bad at discovering new cases of covid-19," 10 April 2020 (Accessed 30 May 2020). [Online]. Available: https://web.archive.org/web/20200426200909/https://www.theverge.com/interface/2020/4/10/21215267/covid-19-contact-tracing-apps-bluetooth-coronavirus-flaws-public-health

[2] R. Raskar, "God's eye view: Will global ai empower us or destroy us," November 2019. [Online]. Available: https://www.ted.com/talks/ramesh_raskar_god_s_eye_view_will_global_ai_empower_us_or_destroy_us

[3] R. Raskar, I. Schunemann, R. Barbar, K. Vilcans, J. Gray, P. Vepakomma, S. Kapa, A. Nuzzo, R. Gupta, A. Berke, D. Greenwood, C. Keegan, S. Kanaparti, R. Beaudry, D. Stansbury, B. B. Arcila, R. Kanaparti, V. Pamplona, F. M. Benedetti, A. Clough, R. Das, K. Jain, K. Louisy, G. Nadeau, V. Pamplona, S. Penrod, Y. Rajaee, A. Singh, G. Storm, and J. Werner, "Apps gone rogue: Maintaining personal privacy in an epidemic," 2020.

Slowing the Spread of Infectious Diseases Using Crowdsourced Data

Sydney Von Arx, Isaiah Becker-Mayer, Daniel Blank, Jesse Colligan, Rhys Fenwick, Mike Hittle, Mark Ingle, Oliver Nash, Victoria Nguyen, James Petrie, Jeff Schwaber, Zsombor Szabo, Akhil Veeraghanta, Mikhail Voloshin, Tina White, Helen Xue

A Introduction

We are a group of volunteers — researchers, software engineers, privacy and public health experts — who have developed a privacy-preserving mobile app intervention to reduce the spread of COVID-19. Our mobile app performs automatic decentralized contact tracing using bluetooth proximity networks.

Our volunteers care strongly about preserving human life and human rights. All data we could collect is voluntary and fully anonymized. All code is transparent. It is open source [1] and could be easily reviewed, reproduced and used anywhere on the planet.

The app could be installed by anyone with a bluetooth-capable smartphone, alerting them to their risk of having been in contact with a confirmed case of COVID-19, and helping them to protect themselves and their friends, families, and other contacts altruistically.

We believe scalable measures like an app are especially helpful in communities where contact tracing resources are too limited to match the scope of the pandemic. We're building this app to provide components and tools that public health agencies can use to supplement their pre-existing efforts to fight COVID-19, assisted by voluntary public action.

A.1 Privacy Focus

Existing mobile apps without a privacy focus have been an effective intervention to reduce the spread of COVID-19. However, invasive interventions carry significant human rights costs, including the temporary loss of personal freedom and fears around whether that freedom will be restored.

A mobile app with a strong privacy model may also have greater efficacy because people will be more likely to share accurate data if they know that data is safe. Ensuring privacy prevents COVID-19 patients from being ostracized or socially harmed on account of inadvertent potential data exposures. Also, mobile apps with poor privacy models may further undermine public confidence in responses and exacerbate existing mistrust.

In contrast, our mobile app research has focused on developing a strong privacy model while still providing effective intervention. Using the app, users can alert recent contacts without anyone being able to trace the information back to them. We believe this intervention has the potential to slow or stop the spread of COVID-19 and save lives.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

A.2 Current Mobile Phone Interventions

South Korea and China have demonstrated two successful systems for containing COVID-19 that make extensive use of technology. The results they have seen match well with predictions from numerical models: with a sufficient diagnosis rate and contact tracing accuracy COVID-19 can be contained.

China [2] was the first to create a mobile app intervention. Their app uses GPS history and other data to assign a risk score. This score is then used to control which individuals are allowed to move freely. China's intervention appears to have been successful, but required far-reaching state surveillance that, by the standards of most liberal democracies, would be considered highly invasive, likely unlawful, and politically unpalatable.

South Korea [3] publicizes a large amount of information collected from the cellphones of infected patients so that others can determine if they had been in contact. South Korea's success has been attributed mostly to (1) widespread testing (2) contact tracing and (3) case isolation. However, their mobile alert solutions do not effectively anonymize patient data. They gather location data from interviews, mobile phone GPS history, surveillance cameras, and credit card records then send text alerts with the location history of patients. Much like the intervention in China, this appears to be effective, but takes a similarly high toll on personal privacy.

We've built a privacy-preserving version of these successful interventions that we believe would have a regulatorily, publicly, and politically viable adoption process within the United States and other Western countries. The system as designed complies with existing regulations around medical information in the United States and does not reveal identifying patient information.

A.3 Making Interventions More Efficient

Non-pharmaceutical pandemic interventions fundamentally make a trade-off between two important social goods: (1) loss of life from the pandemic and (2) economic impact, which influences health and well-being outcomes indirectly. Mobile app interventions are a powerful public health tool because they can improve this trade-off.

In general, non-pharmaceutical approaches to infectious disease control have the following components:

- Filtering (picking a subset of the population)
- Intervention (modifying the behaviour of these people)

For example, quarantining patients with a positive diagnosis applies a filter based on testing and then applies the quarantine intervention. Other examples include travel restrictions for at-risk areas, cancelling public events in a specific city, or encouraging more handwashing in an entire country. Some of these interventions, especially self-isolation, are highly effective [4] at preventing the transmission of infectious diseases like COVID-19. The downside is that they can also be costly to use.

The quality of filtering plays a crucial role in determining the trade-off between loss of life and economic impact. If filtering is poor, a correspondingly larger economic impact will be needed to achieve the same loss of life reduction. Without good filtering, broad quarantines and social distancing are needed, incurring a huge cost in the form of negative impact on people's lives.

Unfortunately, traditional approaches to filtering, such as contact tracing, are labor intensive and don't scale well. So we expect filtering (and, correspondingly, the trade-off between loss of life and economic impact) to degrade in quality as a pandemic grows. But automated contact tracing solutions have the potential to be more scalable – and potentially even more accurate, with access to higher quality information than traditional contact tracing. This may allow for a better trade-off to be maintained in the midst of a pandemic.

B Proposed System: Three Parts

The system proposed here is intended to be used as part of a broader campaign to combat COVID-19 immediately and in the long term. These methods focus on gathering and disseminating the information needed to perform

targeted interventions.

There are three components to this system that work almost independently, but can be bundled into a single mobile app. Depending on privacy requirements and the needs of specific public health authorities a subset of these capabilities could be utilized.

1. Automated contact tracing at scale using anonymized bluetooth proximity sensing
2. Heatmap informed by epidemiological models using anonymized GPS data to warn users about high risk areas
3. Recommendations from local health authorities and risk-aware suggestions about when to get tested

C Part 1: Bluetooth Contact Tracing

C.1 Contact Tracing Background

Non-pharmaceutical methods focused on social distancing reduce the spread of COVID-19. These methods are based on reducing contact between infected and susceptible people, even when it isn't known who is infected. In the simplest form, this is being achieved by reducing all social events and increasing precautions like handwashing. This is an effective measure because the number of new infections is roughly proportional to the number of contact events with infectious people.

A more targeted approach employed by many health agencies is contact tracing [5]. This system works by finding and monitoring contacts of patients that have been diagnosed. Individuals that are thought to be infected are then put into isolation to prevent further transmission, and individuals who have previously been in contact with infected individuals are quarantined.

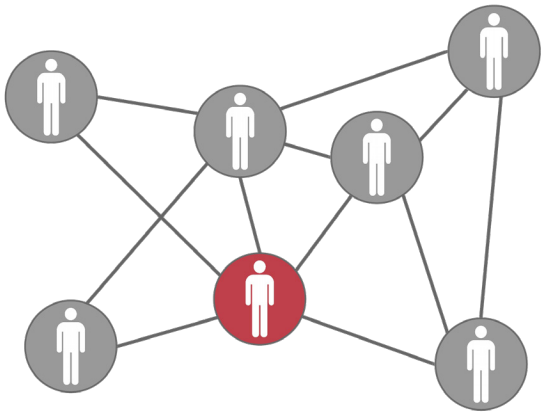


Figure 1: Contact Network

Hellewell et al. [6] analyzed the effectiveness of contact tracing as a method to contain COVID-19 at the beginning of an outbreak. **Their findings are promising: with 80% contact tracing accuracy and a mean detection time of 3.8 days after symptom onset, containment is likely.**

In the context of contact tracing, there are three parameters that models show [7] can strongly impact results:

1. Reduction in overall transmission through social distancing
2. Testing rate and time to diagnosis
3. Contact tracing accuracy

C.2 Model Description

Mobile phones are carried by a majority of people in several countries with an estimated 3.5 billion users worldwide. They are extremely common in Western society, with over 70% of the US population estimated to own one. Bluetooth is a radio protocol that can be used to wirelessly communicate between nearby mobile devices and the signal strength can be used to estimate distance.

Mobile devices can be made to proactively record contact events with other nearby devices by sending bluetooth signals. By measuring the signal strength and discrete number of contact events, the duration and distance of contact between two phone users can be estimated.

By recording all contact events, a high accuracy list of at-risk individuals can be generated automatically when a new person is diagnosed. These individuals can then be immediately notified to ensure they self-isolate before infecting more people.

Bluetooth proximity may be the most accurate crowdsourcing method for approximating close contact to perform contact tracing.

While GPS data is a more well-known general technology, there are significant advantages bluetooth has over GPS in terms of accuracy for contact tracing. With bluetooth, proximity can be approximated by signal strength that is reduced by obstructions like walls; therefore, it more accurately reflects functional proximity in high-risk environments for close contact: inside buildings, in vehicles and airplanes, and in underground transit.

Bluetooth communication also occurs directly between mobile devices. This means a decentralized system can be built more easily with and with stronger privacy protection than other crowdsourcing data types like GPS trajectories.

We are also pursuing research in developing inexpensive external bluetooth devices under the same automatic contact tracing alert system for use in communities with fewer smartphone users. These methods would face much steeper adoption challenges, but if mobile app users and external device users could be integrated under a single system, outcomes could be further improved over more global communities.

C.3 Privacy Model

The bluetooth contact tracing system (Fig 2) can be structured in a decentralized and anonymous way using randomly generated and locally stored 'Contact Event Numbers'. This allows the system to function fully without any private information being stored or transmitted.

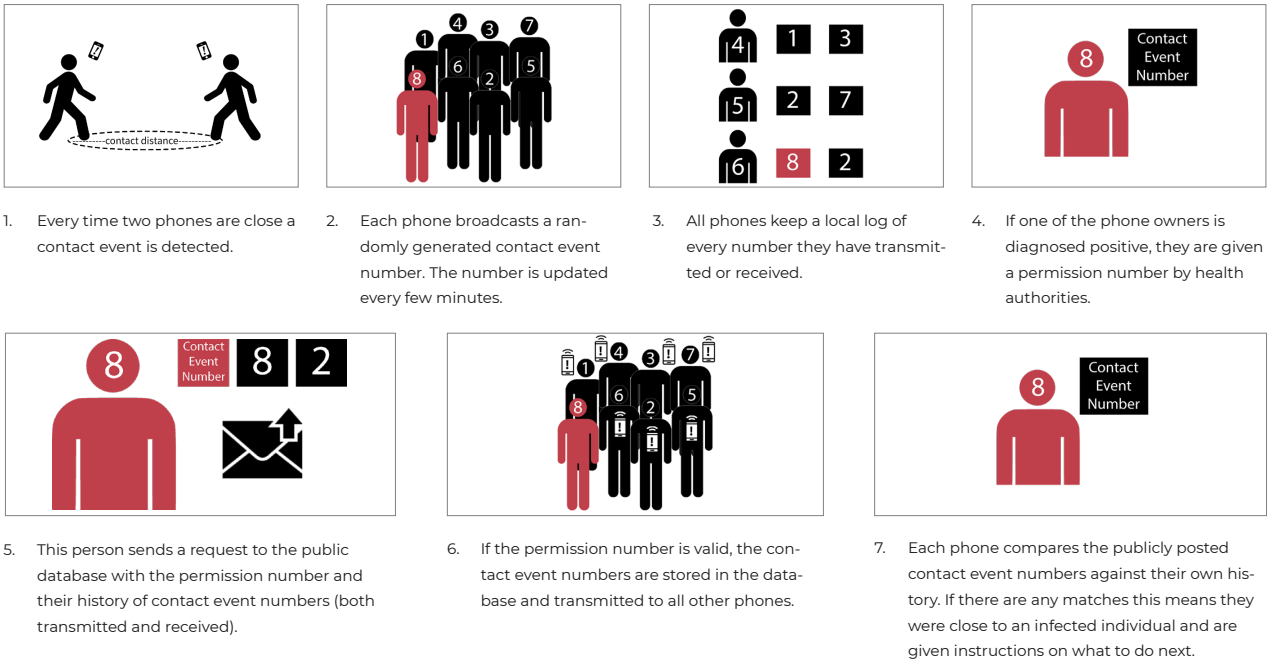
By generating a new random number for each contact event, the system is able to operate without storing or transmitting any personal information. This method is designed so that only the phones involved in a contact event are able to identify messages on a public database.

The only authentication required is the permission number provided by a public health authority. This permission number is used so that malicious actors cannot send false alarms. After authentication, the permission number is deleted from server memory.

The contact event numbers are random and only known by the message recipients and the message sender, so the database can be made public without risk of sensitive information being discovered. While our current intervention is based on permission numbers, in regions where widespread testing is unavailable, a well-designed symptom sharing questionnaire may perform a similar function, with a higher number of false positives. Research in this direction is currently being done by the CoEpi team. The most effective form of this intervention would occur in communities that implement widespread testing and where permission numbers are shared with the mobile app by public health departments.

C.4 Database

The specification for the database is very simple: it is shared across all installations of the app and stores anonymized Contact Event Numbers. If protections against hoaxes are required, permission numbers can be used,



1. Every time two phones are close a contact event is detected.
2. Each phone broadcasts a randomly generated contact event number. The number is updated every few minutes.
3. All phones keep a local log of every number they have transmitted or received.
4. If one of the phone owners is diagnosed positive, they are given a permission number by health authorities.
5. This person sends a request to the public database with the permission number and their history of contact event numbers (both transmitted and received).
6. If the permission number is valid, the contact event numbers are stored in the database and transmitted to all other phones.
7. Each phone compares the publicly posted contact event numbers against their own history. If there are any matches this means they were close to an infected individual and are given instructions on what to do next.

Figure 2: Privacy Model

but they are not required for the privacy model. If the database grows too large, it can also be fragmented based on general location. The code and a more in depth discussion of architecture are available on the open source github repo [1].

C.5 Implementation

Bluetooth contact tracing is implemented via background processes on iOS and Android. The approach currently being investigated utilizes BLE functionality for background advertisement and scanning. Due to different system requirements for Android and iOS, the protocol works differently depending on the operating systems of the devices involved. The key challenges are:

1. iOS devices acting as “peripherals” in the background can only be found by “centrals” that are scanning for their specific service UUID. These peripherals must establish a connection to transfer any data.
2. Android devices have several unfixed bugs where subsequent connections with many devices can cause the bluetooth system to lock up.

The current solution is a hybrid model that is asymmetric for communication between iOS and Android. All devices will simultaneously act as peripherals and centrals, but only some devices will be able to detect others, and only some devices will need to establish a connection to exchange data. An extended description of the communication model and the code are available on our Github repo [1]. This model has been successfully implemented as a proof-of-concept.

D Part 2: GPS Heatmap

D.1 Model Description

GPS capability is ubiquitous among smartphone devices, and high-resolution spatiotemporal data is regularly used by mapping and social media apps. Given anonymised GPS data along with user infection status, we can approximately calculate where the fomite-based risk of infection is the highest; meaning, where there may be inanimate objects capable of transmitting infection. Anonymized GPS data can inform an epidemiological model to show how the disease is likely to spread. Based on this epidemiological model, we can generate a risk heatmap showing comparative fomite risk for different geographical areas at the current time. This heatmap enables our users to adjust their behaviours in response to their local environment. They could, for example, choose to take extra precautions when in high-risk areas, or avoid them altogether. This both decreases the risk of our users becoming infected, and decreases the risk that they will infect those around them. This provides a supplementary service for our users to inform their social distancing measures.

D.2 Epidemiology Model

To describe it simply, the heat map builds upon SEIRS models [8]. The population of simulated users, and probability and timing of a simulated user moving from one state to the next, are generated from the data we have available: both on the background demographics of impacted areas and specific case data. Each simulated user is given an age and has a chance of having pre-existing health conditions, which impact their contagiousness and susceptibility to infection. The anonymised user trajectories are superimposed over a grid covering the area being modeled, while the grid squares are filled with a number of randomly generated simulated humans based on demographic data for that area. As simulated infected and contagious users move across the grid, they carry a chance of infecting the other simulated humans (both user and non-user) sharing their grid space at any given time. Simulated non-user humans also have a probability of adjusting their behaviour and movement throughout the simulation, to model hospital admission and self-isolation. This simulation is run multiple times, and the amount of time each infected human spent in each grid square contributes to its overall risk score, which is represented by the final heatmap. As our understanding of COVID-19 progresses, this heatmap can be refined, and the parameters tuned to more accurately reflect real-world data.

D.3 Privacy Model

Given the personally identifying nature of the spatiotemporal trajectories provided by GPS systems, this information will be handled in a more complex way involving two distinct servers. The first server, Server A, will handle anonymisation using the methods described in the following GPS Anonymization Model section. Users will send their GPS data to Server A first without any other information, and Server A will return the now-anonymised data to them. They will then upload the anonymised data, along with their infection status, to the second server, Server B. Server B will take this data and add it to our epidemiological simulation, generating a heatmap.

D.4 GPS Anonymization Model

The heatmap may either require the application of an anonymization algorithm, or the explicit consent of users to publish high resolution GPS trajectories. There is a large body of academic work [9] behind the anonymisation of spatiotemporal data, most of which draw from a handful of common strategies. These include aggregating multiple similar trajectories together, decreasing the resolution of each datapoint across space and time, removing particularly distinctive datapoints, and swapping sections of trajectories that cross paths.

Of the several metrics used for quantifying de-identification methods, we have chosen k-anonymization. A dataset is anonymous for some integer k if no entry can be narrowed down further than belonging to one of k

individuals. For example, a group of trajectories would be 3-anonymous if each trajectory could plausibly belong to at least 3 individuals. Our aim is to create a dataset with the maximum possible k-value that still preserves utility. The worst-case scenario we would be willing to publish is at minimum a 10-anonymous dataset with respect to an adversary using commonly available geographical information.

Given the urgency of the situation, we intend to use contractual boundaries against deanonymization rather than aiming to protect against more sophisticated and hypothetical attacks on our anonymization scheme. Building anonymity systems that are resistant to more sophisticated attacks is challenging, and takes much more time.

The anonymisation model we are implementing is one created by [10]. It allows for flexible k-anonymity without significant distortion, and has a proven and open-source implementation. This does require a significant number of trajectories to be accurate, so we will be supplementing our initial database with plausible synthetic trajectories created by the Brinkhoff trajectory generator. The end result is that the trajectories being used to generate our heatmap can be mathematically verified to be anonymous, guaranteeing the privacy of our users.

E Part Three: User Recommendations

We have designed the app user interface (UI) and are building a beta app with the following features:

- CDC general COVID-19 advice, symptoms, and resources
- An infection density heat map based on anonymized GPS data
- A notification system of potential COVID-19 contact risk via bluetooth proximity networks
- Personalized advice: If close contact is detected, a popup instructs the user to call the local public health department, and looks up this number to call for them to inquire about next steps
- Future features may include: more personalized advice based on heat map location, a supplies map, self-reporting of symptoms, FAQs, news updates tailored by geographic region, travel counseling, or access to home-based testing

F Why You Should Care

Incentives for Health Authorities

- High accuracy, instantaneous contact tracing
- Targeted interventions based on calculated risk and current health policy
- Easier communication of announcements and information

Incentives for Individuals

- Information about how to avoid contracting the disease
- Earlier warning to protect friends, family, and close contacts if you do get sick
- Friends, family, and close contacts who use the system will likely be warned before they can get you sick
- Broad health measures to contain the disease could be relaxed in favor of targeted and private interventions, so regular life will not need to be disrupted as much

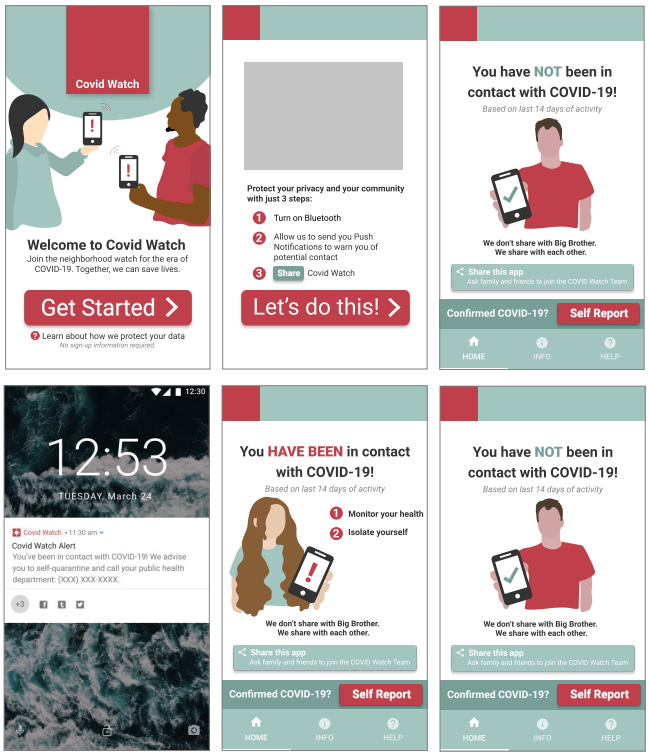


Figure 3: **(top)** User Interface Design: New User OnBoarding Workflow. Downloading and using Covid Watch does NOT require any sign up of email, password, etc. of any kind. The only requirement is to enable Covid Watch to access bluetooth on your smartphone in order to detect other smartphones in close proximity to log a 'contact event'. If no other smartphones you have been in contact with are associated with a positive case COVID-19, Covid Watch informs you that you have not been in contact with COVID-19. **(bottom)** Contact Alert and Reporting. If Covid Watch detects another smartphone within bluetooth proximity that is associated with a positive COVID-19 case, you are alerted that you may have been exposed Covid Watch suggests steps of (1) monitoring your health and (2) isolate yourself. Additionally, you may update your own status as confirmed or not tested, along with the first date of symptoms.

F.1 Quantitative Analysis of Impact

The impact of this technology will depend largely on the state of the world around it. Numerical models [7] and ongoing campaigns [3] suggest that with extensive testing, accurate contact tracing, and isolation of suspected cases, outbreaks can be contained.

For intermediate testing and contact tracing detection rates, a system like this would likely need to be used in combination with continued social distancing measures and manual contact tracing. However, the measures suggested by Ferguson et al. could potentially be relaxed if supplemented with sufficient targeted interventions.

Of the three target parameters (tracing accuracy, detection rate, base transmission), the potential influence of the app on tracing accuracy is the simplest to quantify. Any two users of the app who are at the same location at the same time will register a contact event. In theory, all transmission events except those by fomites would be detected. This includes all types of contact classified as being "high risk" by the ECDC.

Preventative measures encouraged by the app such as avoiding high-risk areas and increased precautions would reduce overall transmission rate, however it is difficult to quantify this impact. The current design of the app may also increase detection rate by informing users of symptoms to watch for and how to get tested. Ongoing research is being conducted on how to allocate testing resources to maximize the expected value of secondary

cases detected.

To have a significant impact the technology will require a significant adoption rate. To detect contact events using bluetooth both individuals must have the app installed at the time of transmission. Assuming P percent of the population uses the app, A percent of transmission events between app users are detected, and T percent of infected are tested, then TAP^2 percent of total transmissions are detected (if app users are homogeneous in the population).

Figure 4 below examines the percentage of detectable transmission events in countries with different testing rates as a function of app usage. The analysis assumes that these events are uncorrelated, which may significantly under-estimate detected transmissions due to missing chain-reaction testing. Ongoing modeling work is being done to investigate the dynamics of rapid tracing and testing along infection chains.

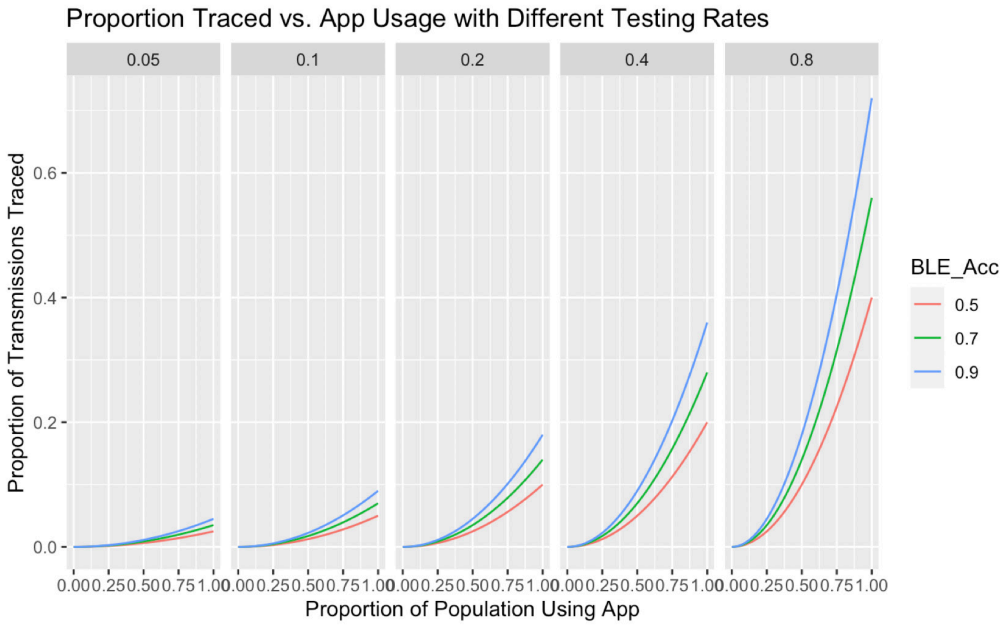


Figure 4: Transmission Detection vs. App Usage curves for testing rates $[0.05, 0.1, 0.2, 0.4, 0.8]$. BLE_Acc gives the detection rate of transmission events between app users.

In order to investigate disease dynamics with clusters of app users, the model developed by Hellewel et al was modified to account for two sub-populations, one that uses the app and another that doesn't. The modified implementation is available on Github here and currently assumes 90% app tracing accuracy and 50% non-app tracing accuracy (assuming health systems are partially overwhelmed).

Figure 5 was used to estimate the average number of infections caused in 6 weeks due to a single imported case. Results show a significant reduction in risk for clusters of app users that are partially distanced from non app users. This is important because it provides an additional incentive for individuals to use the app even when the overall adoption rate is low. As the proportion of the population using the app increases the risk for the entire population is reduced and most outbreaks are contained.

What we've most wanted to know is the answer to this question: "Can an effective contact tracing program reduce local transmission so that sustained local spread does not occur?" The answer looks like yes. With a comprehensive testing program, high mobile app contact tracing accuracy, and self-isolation of diagnosed individuals, our models predict that each new case could cause on the order of 10 other cases before the outbreak is extinguished. Also, even in parameter regimes where automated contact tracing alone is not enough, this technology can be used in combination with existing methodologies to provide greater protection with lower

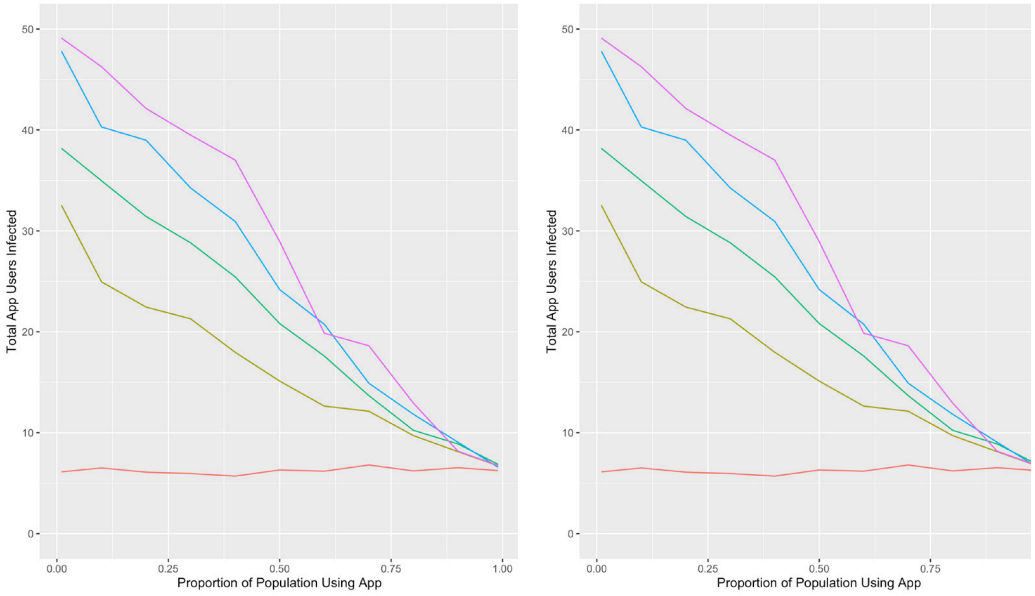


Figure 5: Expected infections for total population (1st) and app population (2nd) adjusted for relative population size.

social cost.

G Timeline

We estimate we could complete the remainder of the necessary technical requirements and deploy the first version of the mobile app for pilot within 2 weeks, and possibly sooner, based on the rapid rate of progress that has so far been achieved by the researchers and volunteers who have devoted their time to this technically sophisticated, but potentially high impact intervention.

The track to development requires (1) more volunteers to help us with skill sets listed on our collaborate page and (2) funding for, at a minimum, cloud services.

The app will be implemented and launched in two parts:

1. The first version will implement the Bluetooth proximity network and risk heatmap to notify users of potential close contact exposure to SARS-CoV-2.

2. The second version will build upon the initial launch. Users may be able to self-report symptoms to receive personalized advice based on their risk score and advice from the local health department.

We intend to build the tools for all of these systems, but understand that regulatory requirements may vary by region. The app will be designed so that each component can be used separately if necessary.

We also want to emphasize that high user adoption as quickly as possible after release will facilitate the best possible outcomes for intervention. The app may not be able to make use of historical GPS data from individuals, and historical data from bluetooth contact events doesn't exist. Therefore, a user needs to download the app in order to start the clock and begin benefiting from the system as it anonymously logs bluetooth contact events.

H Conclusions

Mobile technologies can provide instantaneous and high accuracy contact tracing, even between strangers at low social and economic cost. Instead of requiring thousands of healthcare workers to do this manually, the process will be essentially cost-free. Because the system will be so accurate, a majority of people can resume living their lives normally without the need for indefinite social distancing.

We are developing this technology as a high-quality filter to be used for the pandemic optimization problem. Combined with a comprehensive testing program, our filter may be powerful enough to stop further spread of COVID-19.

Who Can Help We continue to seek more collaborators, donors, and support from health organizations and testing centers.

References

[1] *Covid Watch*, (Accessed 30 May 2020). [Online]. Available: https://github.com/covid19risk/

[2] D. G. McNeil, "Inside china's all-out war on the coronavirus," *New York Times*, Mar 2020. [Online]. Available: https://www.nytimes.com/2020/03/04/health/coronavirus-china-aylward.html

[3] D. Normile, "Coronavirus cases have dropped sharply in south korea. what's the secret to its success?" *Science Magazine*, Mar 2020. [Online]. Available: https://www.sciencemag.org/news/2020/03/coronavirus-cases-have-dropped-sharply-south-korea-whats-secret-its-success

[4] N. M. Ferguson, D. Laydon, G. Nedjati-Gilani, N. Imai, K. Ainslie, M. Baguelin, S. Bhatia, A. Boonyasiri, Z. Cucunuba, G. Cuomo-Dannenburg, A. Dighe, I. Dorigatti, H. Fu, K. Gaythorpe, W. Green, A. Hamlet, W. Hinsley, L. C. Okell, S. van Elsland, H. Thompson, R. Verity, E. Volz, H. Wang, Y. Wang, P. G. Walker, C. Walters, P. Winskill, C. Whittaker, C. A. Donnelly, S. Riley, and A. C. Ghani., "Report 9 - impact of non-pharmaceutical interventions (npis) to reduce covid-19 mortality and healthcare demand," Mar 2020. [Online]. Available: https://www.imperial.ac.uk/mrc-global-infectious-disease-analysis/covid-19/report-9-impact-of-npis-on-covid-19/

[5] *Public health management of cases and contacts associated with coronavirus disease 2019 (COVID-19)*, 10 April 2020 (Accessed 30 May 2020). [Online]. Available: https://www.canada.ca/en/public-health/services/diseases/2019-novel-coronavirus-infection/health-professionals/interim-guidance-cases-contacts.html

[6] J. Hellewell, S. Abbott, A. Gimma, N. I. Bosse, C. I. Jarvis, T. W. Russell, J. D. Munday, A. J. Kucharski, W. J. Edmunds, F. Sun, S. Flasche, B. J. Quilty, N. Davies, Y. Liu, S. Clifford, P. Klepac, M. Jit, C. Diamond, H. Gibbs, K. van Zandvoort, S. Funk, and R. M. Eggo, "Feasibility of controlling covid-19 outbreaks by isolation of cases and contacts," *The Lancet Global Health*, vol. 8, no. 4, pp. e488–e496, 2020/05/31 2020. [Online]. Available: https://doi.org/10.1016/S2214-109X(20)30074-7

[7] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, M. Parker, D. Bonsall, and C. Fraser, "Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing," *Science*, vol. 368, no. 6491, 2020. [Online]. Available: https://science.sciencemag.org/content/368/6491/eabb6936

[8] *SEIR and SEIRS models*, Accessed 30 May 2020. [Online]. Available: https://www.idmod.org/docs/hiv/model-seir.html

[9] V. Primault, A. Boutet, S. B. Mokhtar, and L. Brunie, "The long road to computational location privacy: A survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, p. 2772–2793, 2019. [Online]. Available: http://dx.doi.org/10.1109/COMST.2018.2873950

[10] M. Ghasemzadeh, B. C. Fung, R. Chen, and A. Awasthi, "Anonymizing trajectory data for passenger flow analysis," *Transportation Research Part C: Emerging Technologies*, vol. 39, pp. 63 – 79, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X13002623

CoVista: A Unified View on Privacy Sensitive Mobile Contact Tracing Effort

David Culler, Prabal Dutta, Gabe Fierro, Joseph E. Gonzalez, Nathan Pemberton,
Johann Schleier-Smith, K. Shankari, Alvin Wan, Thomas Zachariah
{culler, prabal, gt.fierro, jegonzal, nathanp}@berkeley.edu
{jssmith, shankari, alvinwan, tzachari}@berkeley.edu
UC Berkeley

Abstract

Governments around the world have become increasingly frustrated with tech giants dictating public health policy. The software created by Apple and Google enables individuals to track their own potential exposure through collated exposure notifications. However, the same software prohibits location tracking, denying key information needed by public health officials for robust contract tracing. This information is needed to treat and isolate COVID-19 positive people, identify transmission hotspots, and protect against continued spread of infection. In this article, we present two simple ideas: the ***lighthouse*** *and the* ***covid-commons*** *that address the needs of public health authorities while preserving the privacy-sensitive goals of the Apple and google exposure notification protocols.*

A Introduction

Apple and Google have adopted a decentralized approach to mobile contact tracing that prioritizes individual privacy [1]. Under the Apple-Google Exposure Notification (AGEN) protocol (see Fig. 1), individual phones determine if the user has been exposed, without revealing *the identity of the infected individual* and *where the contact event took place* The AGEN protocol is related to contemporaneously proposed protocols including PACT and DP-3T [2, 3]. Like these other protocols, the AGEN protocol does not use location information. Instead, it relies on the Bluetooth radios present on all modern phones to detect proximity with others. Beyond not collecting Protected Health Information (PHI), the decentralized approach retains the non-PHI on the phone, allowing individuals to determine risk locally on their device.

Governments and public health authorities want to understand where and how the disease is spreading, so they can take preventative measures. They also want to be able to use mobile contact tracing to augment existing manual contact tracing efforts. With these goals in mind, governments advocate for a centralized approach, whether national or regional, where they maintain records of each person's locations and interactions. This allows governments to determine exposures and notify people directly, as timeliness reduces spread. While centralized contact tracing may offer utility critical to re-opening the world's economy, it raises profound concerns for civil

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

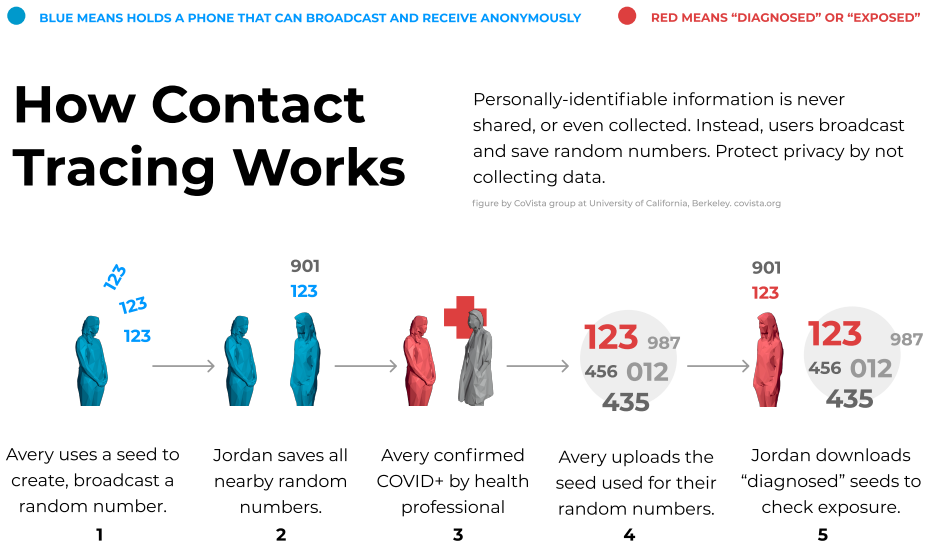


Figure 1: Apple-Google Exposure Notification (AGEN) Protocol Overview.

liberties and personal privacy. Government efforts that avoid reliance on the industrial Exposure Notification offerings have run into a host of failings, including reliability, power drain, interoperability, and participation.

Apple and Google have taken an unprecedented position – essentially dictating public policy, not just by requiring the decentralized approach, but also by prohibiting contact tracing apps from collecting location information. Further, they are restricting access to the new contact tracing APIs to national governments and permitting only one app per country or region. This decision circumvents the local governments, tribal organizations, and community health services that are often most aware of existing manual contact tracing efforts and the needs of their communities. Meanwhile, government contact tracing apps have failed due to restrictions imposed by AGEN.

In this article, we present two simple measures that enable the AGEN protocol to support manual contact tracing efforts, provide visibility into the spread of disease, and return authority to local communities all while preserving privacy within the Apple and Google framework.

1. **Treat places as people.** Endow public places with the same privacy-preserving technology used to monitor exposure for individuals.

2. **Nation-scale data, not apps and processes.** Build a common backend for the AGEN protocol that spans apps and governmental boundaries.

In the rest of this article, we describe these two simple measures and how they both improve contact tracing while also preserving individual privacy.

Lighthouse: Treat Places as People

If we treat public places as people, we can use the AGEN protocol to (a) understand COVID-19 exposures across space, (b) integrate with manual contact tracing, and (c) do so with the same privacy-sensitive protocol. To treat places as people in AGEN, simply attach mobile phones or specialized low-cost beacons to publicly accessible places (e.g., county services, stores, buses). Like a lighthouse, these devices help communicate risk associated with places. Well-positioned, they can offer robust proximity detection, can detect their exposure, and can convey aspects the risk that represents.

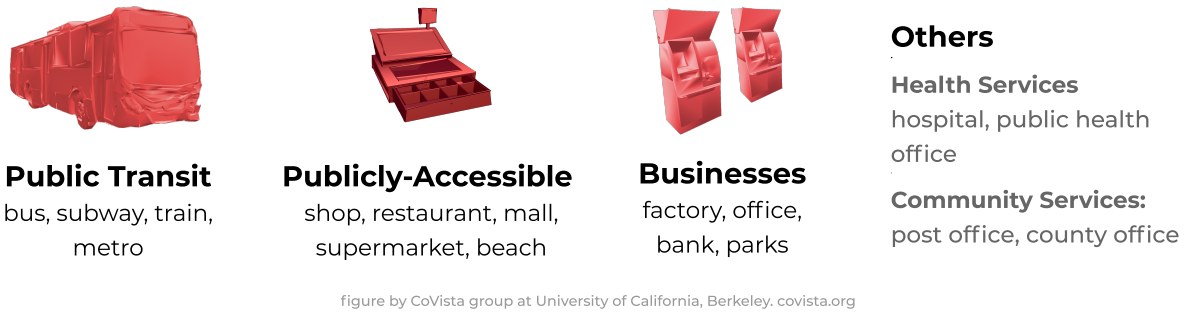


Figure 2: Lighthouses can extend the AGEN protocol to physical places

By choosing to share their locally computed exposure risk with public health authorities through the AGEN protocol, owners of publicly-accessible places can aid in mitigating virus spread. Alternatively, if a place is identified through traditional, manual contact tracing, the place can still anonymously participate in the AGEN protocol, notifying others without revealing where they were exposed. Treating places as people empowers stewards of public spaces to collaborate with public health authorities to help mitigate the spread of disease without jeopardizing the privacy of patrons or the reputation of the public spaces. This procedure can facilitate detection of exposure from a non-participating individual while improving anonymity over manual contact tracing methods. Going even further, such places could provide other means of beaconing that do not involve smartphones, such as QR code displays, codes on receipts and so on.

COVID Commons: A Nation-scale Data Backend

Rather than "one app per nation," a better solution would be to provide a common privacy-preserving data exchange across apps and administrative boundaries — a Commons. This would allow societal structures and innovation, rather than corporate policy, to determine how the app ecosystem should evolve. It is very likely that participation will be greatest if the apps are available through local organizations (e.g., tribal organization, university campus) that individuals trust. A common privacy-preserving data exchange is already compatible with the AGEN protocol.

When an individual tests positive and they engage in a conventional contact tracing interview with a public health professional. The professional obtains an authorization so the individual, on an opt-in basis, can share their anonymous exposure information. Public health professionals serve to protect the integrity of the information in the Commons without exposing any patient data or medical data.

Their actions are quite similar to publishing counts of cases, statistics and demographic information, as is done today. The Commons might be hosted by governmental or NGO structures, based on national or regional policy. A diverse and innovative app ecosystem can grow to meet the needs of individuals and agencies.

In the remainder of this article we describe both these technical solutions in greater detail. We have organized each section to be relatively self-contained.

B Background on Privacy-Sensitive Mobile Contact Tracing

The key building block for Privacy-Sensitive Mobile Contact Tracing (PS-MCT) is a subtle combination of radio protocols, cryptography, and risk-calculation. Phones have a short-range radio, Bluetooth, used to connect to nearby devices. To make those connections it periodically broadcasts tiny bits of information. The PS-MCT protocols leverages this short-range background broadcast to resolve nearby individuals.

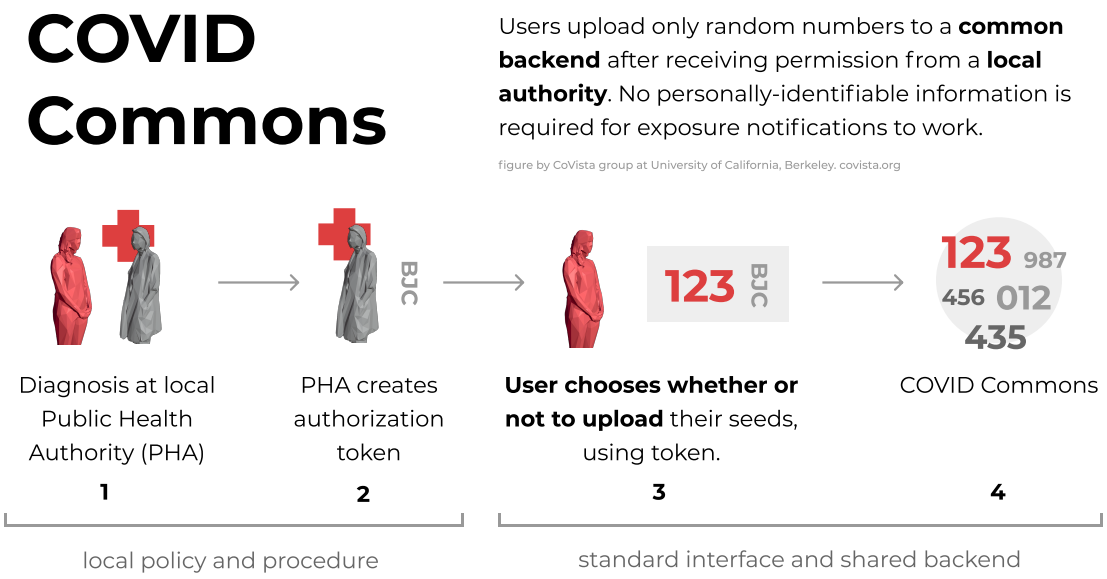


Figure 3: Covid Commons exposure notification process.

In the Apple-Google Exposure Notification (AGEN) protocol, each phone generates a daily secret key called a Temporary Exposure Key (TEK). Then every 15 minutes the phone uses the TEK to generate a new 16 byte Rolling Proximity Identifier (RPI). The RPI sequence is generated using a cryptographic hash function, so it does not carry any information about the source individual. The current RPI is then continuously broadcast every few hundred milliseconds. All phones log the RPIs they hear for future exposure analysis. Because the RPI is continuously changing, it also cannot be easily tracked.

When someone tests positive they can **anonymously** publish the daily keys (TEKs) from the days when they were contagious. The confirmed positive collection of TEKs is called a Diagnosis Key in the AGEN protocol. The Diagnois Keys are published by sharing them with a trusted server which publishes the TEKs for download. Others can obtain these keys and use the same cryptographic hash functions to recreate the sequence of RPIs. This sequence, combined with some region-specific weights, can determine if they encountered any infected individuals. This entire process is accomplished within the Android and iOS operating systems. Government sanctioned apps are only responsible for authenticating infected individuals and, with user permission, publishing the keys.

It is important to note the distinction between policy and mechanism. The AGEN protocol (and the extensions proposed in this paper) provide a mechanism to detect and notify users about exposure risk, but it is up to public health authorities to define what constitutes an exposure. This distinction is explored in more detail in section D.

C Lighthouses: Treating Places as People

Despite their promise, there remains significant concern around the efficacy of privacy-preserving contact tracing and exposure notification protocols. For one thing, not everyone has access to a mobile phone, and many that do may be unwilling to participate (even Singapore only achieved 20% adoption[4]). This is a serious concern as the probability of a successful detection grows quadratically with participation [2]. Of course, manual contact tracing does not have this issue and remains the gold standard. Will these two techniques exist in isolation or will they interact synergistically? Finally, bluetooth contact detection is limited in range and time; it cannot detect many

important forms of transmission like surfaces or HVAC systems[5].

C.1 Treating Places as People

There is a simple extension to mobile contact tracing that can help address these issues. The idea is rooted in centuries old maritime signaling. To navigate at night, ships use signal lights, much like our Bluetooth beacons, to identify and safely navigate around nearby ships. However, to safely sail at night, ships also rely on lighthouses, strategically placed beacons, to identify and safely navigate around key landmarks. It is this second form of beacon, the lighthouse, that is needed to address several of the key limitations in privacy-sensitive mobile contact tracing.

C.2 What is an Exposure Notification Lighthouse?

A privacy-sensitive exposure notification lighthouse is a device (e.g., a mobile phone or even a smart sticker) deployed in a public space following the same privacy sensitive mechanisms as individuals. Figure 4 gives a few examples of what lighthouses could look like. Much like the maritime lighthouse, contact tracing lighthouses can be used to inform others of potential exposures associated with public spaces discovered through manual contact tracing. A contact tracing lighthouse can also log passing beacons to inform owners and public health authorities of exposure risks. However, because the lighthouse follows the same privacy-sensitive protocols as individuals, it retains all of the privacy guarantees of the existing protocols. Moreover, by installing contact tracing lighthouses in participating public spaces ranging from stores and restaurants to schools and buses, we introduce a privacy sensitive mechanism to bridge manual contact tracing with mobile contact tracing. Such a bridge gives public health authorities the ability to gain visibility into the spread of disease while preserving individual privacy.

(a) **Existing Devices:** Easily deployed immediately but expensive to deploy to new places. They can also be unreliable when unattended.

wikimedia commons © Travelarz CC-ASA 3.0

(b) **Dedicated Devices:** Cheap and reliable. They will require new development and can be tricky to place correctly.

(c) **No Device:** Allows those without the app or even a mobile device to participate. They may be easier to re-identify and require manual effort from users to check.

Figure 4: Lighthouses can be deployed using a number of different devices with varying trade-offs. Each of these examples is feasible to deploy in the near future.

C.3 What Do We Get From Lighthouses?

Figure 5 walks through an example of how lighthouses could work in a typical case. Let's explore some of these benefits in more detail.

C.3.1 Bridging Place and Mobile

The contact tracing lighthouse empowers stewards of public spaces (e.g., shop owners, school administrators) to collaborate with public health authorities to help mitigate the spread of disease without jeopardizing the privacy of patrons or the reputation of the public spaces. When an individual is tested and confirmed, the manual contact tracing interview process begins. One of the first questions asked is "Can you recall where you have been that might have exposed others?". A place, unlike an individual, has a large set of explicit relationships with its institutional environment - business license, health department approvals, chamber of commerce, etc. The interview process routinely seeks to gather information about places. But often there is little the place can do to help. With its lighthouse, it has a very simple way to provide assistance without undue impact on its reputation. Just like a person, it can check its own potential exposures. And, like a person, it can anonymously publish its own random numbers as COVID positive so that people can use them for detecting their exposure risk. This is especially useful if the individual who tested positive was not using an app that broadcasts their own sequence of random numbers. Unlike a person's phone, stationary lighthouses can be carefully positioned to avoid issues like weak or unreliable broadcast that mobile phones can face. Finally, the public place may also take other measures in assisting its local health authority in detecting hot spots, such as informing them of the exposure risk that it observes.

C.3.2 Indirect Transmission

The lighthouse can be used to address forms of indirect transmission that are not captured in the existing privacy-sensitive mobile contact tracing efforts. If a COVID-positive individual enters a bus and touches or sneezes on several surfaces they could potentially infect others over the next several hours, long after they have left the bus. Air conditioning systems have also been implicated in spreading the virus over large distances [5]. Existing wireless protocols will fail to capture these forms of transmission since they rely on close, contemporaneous radio proximity with the positive individual. However, with the introduction of the lighthouse, the bus or restaurant can automatically determine that it was exposed and choose to anonymously publish its own random number sequence.

C.3.3 Beyond Mobile Contact Tracing

Finally, the lighthouse can also extend the privacy-sensitive mobile contact tracing protocol beyond the smartphone. Places provide channels of communication to individuals that complement smartphones. Simple lighthouse codes might be included on printed receipts or handed to customers. A person without contact tracing on their phone might use such code to manually perform the exposure checks that are automated by the apps. For example, they could enter such codes that they have received into a public website to determine their exposure risk.

C.4 Lighthouse Privacy

There are legitimate privacy interests for places, as there are for people. Businesses may fear irrational boycotts or loss of reputation, and there are risks of perpetuating prejudice or stigma against neighborhoods or ethnic groups (already a growing problem [6]). The good news is that lighthouses inherit the privacy protecting characteristics of the AGEN protocol. In their simplest form, they are phones running ordinary apps, different only because they live in a fixed place rather than in purse or pocket.

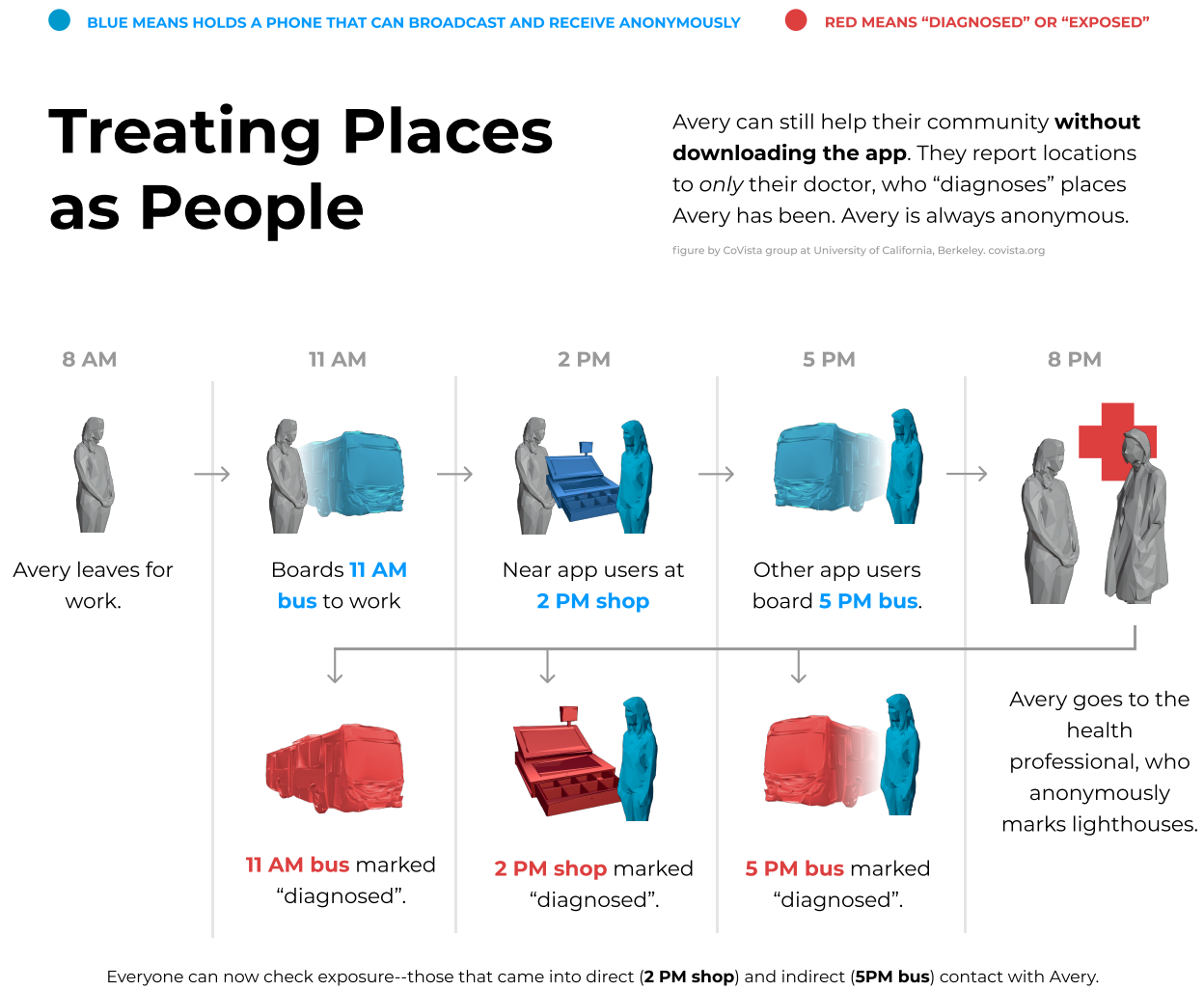


Figure 5: An example scenario of an affected individual (Avery) and a stranger (Bernie) that may have been exposed. Even if Avery is not an app user, the doctor can still notify the shop and bus of their exposure; lighthouses **connect manual and mobile contact tracing**. Lighthouses **bridge place and mobile** by allowing the bus and shop to notice their exposure and mark themselves as exposed, even if Avery did not remember visiting them. They also **enabled less direct forms of exposure** by notifying Bernie of their potential exposure, even though they were not on the bus at the same time as Avery. Finally, lighthouses offer places benefits **beyond the standard mobile contact tracing**. They can now track how often they are exposed, at what times, and even in which section within them (if they have multiple lighthouses installed).

Lighthouses can complement manual contact tracing by offering greater confidentiality in situations where people cross paths in stigmatized locations. For example, investigators following up on a recent case cluster associated with nightclubs in Korea catering to the LGBTQ community have struggled to contact attendees who are afraid of being "outed" or facing discrimination [7]. Lighthouses can fill such gaps without requiring or relying on anyone, even an affected person, to disclose their location history.

A potential concern about lighthouses is that internal state, if combined and aggregated, could be used to undermine some of the privacy guarantees of AGEN. Indeed, with a sufficiently large deployment of lighthouses and centralized collection of recieved RPIs, it would be possible to resolve the locations of all the COVID positive individuals. We therefore stress that each lighthouse must follow the existing decentralized protocol for risk assessment in which only risk is aggregated and not the underlying RPIs.

To address some of these security concerns, it is possible to deploy **transmit only** (passive) lighthouses which could be used to anonymously notify others of exposure without listening for RPIs. This eliminates the risk of resolving COVID positive individuals locations but also limits visibility into where the disease is spreading.

D Covid Commons: Unified Contact Tracing With Many Apps

Standardizing on an exposure notification protocol such as AGEN is crucial to achieving the critical mass of participation required to make such an approach effective (estimated to be at least 80% for a city of 1 million individuals [8]) and privacy preserving. Apple and Google, as producers of the two dominant smartphone OSes, are uniquely positioned to bring these capabilities to nearly every smartphone and have engineered the AGEN protocol to that end. Providing an implementation of the protocol through an OS upgrade would eliminate the fragmentation of the protocol preventing sufficient adoption. However, standardizing the proximity detection protocol is just part of the solution.

Due to concerns over potential misuse of the capabilities of the proximity detection protocol and location tracking features of modern smartphones, Apple and Google have announced that access to the SDK for AGEN to "public health agencies" [9]. The realization of this policy has evolved to essentially mean "one app per state." Centralizing the apps around state governments means that tracing efforts are not limited to local jurisdictional boundaries. However, the public health authorities responsible for testing and tracing often operate at a more local level [10, 11].

The fundamental issue with this policy is a failure to distinguish between ownership of the required repository of "diagnosis keys" and the applications that produce and utilize them. The repository contains keys that correspond to times when diagnosed individuals may have interacted with others and must therefore be public-facing so that individuals, via AGEN-compatible apps, can download the keys and determine their own exposure risk. The keys uploaded to the repository must be limited to those that have been authorized by a public health authority through some testing and interview process. In the U.S., public health measures such as contact tracing and case reporting are carried out at a county or city level through individually executed processes, in coordination with state and national law and policy. However, health authorities do not generally have the technical, human or financial resources to produce the required app and also stand up and maintain such a public-facing data service. This is even less tenable during a pandemic, when such resources are already strained. Who, then, manages the AGEN backend needed to share Report Keys?

The worst-case scenario is for the repositories to emerge as fragmented silos of diagnosis keys. In the midst of the custody battle over administration of the emerging technical approaches to the contact tracing problem, it is important to remember that exposure does not respect administrative boundaries. Regardless of how the repository is established, it is essential that exposure notification can be transmitted across backends and across apps. Indeed, there are already efforts to combine contact tracing efforts and data across the administrative boundaries established by industry's policy around AGEN [12].

D.1 A Path to Nation-scale Data

A solution to this issue is the creation of a privacy-preserving data exchange shared and accessible across apps and administrative boundaries — a Commons. Different public health authorities could cooperatively contribute to the Commons and individuals across many jurisdictions would access it through their apps. Such a Commons might operate at the scale of one per nation, or it might be regional with some form of federation to share keys and exposure information across individual instances. The Commons may be hosted and maintained by governmental authorities, foundations or other appropriate institutional entities.

The Commons requires no change to the EN protocols. Each individual participates in the proximity detection using daily TEKs and the RPIs generated from them without change. With no other information leaving the phone, the app on an indivual phone downloads the diagnosis keys and presents them to the SDK to obtain exposure risk scores. The difference is that those diagnosis keys are not *a priori* limited to the ones produced by users of the same app. Just as in the current decentralized protocol, an individual who tests positive — i.e. a confirmed case — and participates in contact tracing with a specific public health authority is asked to voluntarily submit their TEKs for the past period over which they are likely to have been contagious. The AGEN protocol does not stipulate how that request is formulated or how that submission is performed, but the introduction of the Commons allows us to answer that question precisely.

D.2 The PHA Experience: Federating Access to the Commons

A Commons is utilized by a well-defined set of public health authorities that are registered with it. Professionals at PHAs can authenticate to and access the Commons using well-established authentication techniques such as LDAP and OAuth. As part of requesting a patient to submit their diagnosis keys to the Commons, the professional requests a one-time authorization (OTA) from the Commons. The OTA authorizes the upload of TEKs corresponding to the extent of time during which the patient is likely to have been contagious; this will involve some days prior to the diagnosis and extend to several days or weeks following. The OTA is provided to the patient to authorize the submission of their keys to the Commons; the upload is opt-in, as under the usual AGEN protocol.

Because the OTA corresponds to a single interaction between a healthcare professional and a patient, it is a natural key on which helpful metadata can be associated. The authorizing PHA may want their identity to be associated with the submitted diagnosis keys so that the provenance of the diagnosis is preserved. This identity, realized by the TLS public key of the PHA, can be associated with generated OTAs in the Commons. This provides no more information than having each PHA host their own exposure key store, as currently envisioned. As we will discuss below, the PHA identity can also be used to filter which diagnosis keys are considered in the matching process on an individual's phone.

The OTA itself is also a useful piece of metadata when associated with the uploaded keys in the Commons. The authorizing professional will likely want to be able to determine whether the patient has in fact contributed their keys as requested to determine if follow-up is necessary. Associating the OTA with the uploaded keys places no new information in the Commons: the professional knows what they know about those they interview and they know what authorizations they have requested. Each such confirmed case can be expected to have generated some authorization request, but that information, along with all other aspects of the contact tracing process, is under the purview of the PHA. Counts of confirmed cases and other aggregate information are already regularly reported to the public.

The Commons is able to provide this functionality despite no patient data — indeed, no health or medical data of any kind — ever being entered into the Commons. The Commons also takes no position on which parts of the contact tracing process are automated and which are manual. It integrates cleanly with the existing processes established by PHAs for interviewing patients, deciding which diagnosis keys are appropiate to share, and so on. The Commons merely provides a means of carrying out the result of these decisions in a way that does not

require individuals to be using the same phone app.

D.3 The User Experience: Federating Downloads from the Commons

The current draft of the AGEN protocol says very little about how exposure notification information will be transmitted or relayed among the different instances of the diagnosis key repository. A unified Commons immediately addresses this issue — all diagnosis keys are uploaded and downloaded from the shared repository. However, under a federated regime, it is necessary to address how exposure notification and uploaded keys can be routed across instances of the Commons.

Consider a scenario where an individual is diagnosed by one PHA but may travel or may have traveled to regions covered by different PHAs. Individuals in those other regions should be able to obtain the TEKs for the diagnosed individual, even though the diagnosis was performed by a remote PHA. There are two complementary approaches.

The first approach proactively forwards diagnosis keys to relevant PHAs. A PHA professional can "tag" a requested OTA with public keys or other metadata identifying relevant remote PHAs, using information acquired as part of the interaction with the individual. The process of uploading diagnosis keys authorized by that OTA to the Commons can then incorporate a simple forwarding mechanism that replicates those keys to the federated instances of the Commons managed by those other PHAs. This can be performed by the Commons itself, or it may be performed on an opt-in basis by the individual.

Under the second approach, individuals may proactively request diagnosis key downloads from federated instances of the Commons they are interested in. An individual may subscribe to diagnosis key downloads for Commons covering regions the individual has travelled to or will travel to. Additionally, an individual may subscribe to Commons for surrounding regions.

Regardless of whether the Commons is realized as an administratively centralized server or a federated mesh, the Commons must provide some mechanism for filtering or scoping the download of diagnosis keys onto an individual's device. The number of keys downloaded from the Commons will grow with the adoption of AGEN-based apps, improvements in testing, and the spread of COVID-19. Without the ability to filter the set of keys down to a reasonable and relevant set that is still large enough to maintain the privacy-preserving properties of AGEN, the bandwidth requirements and compute requirements (for deriving the thousands of RPIs used for the matching process) may grow to be untenable.

D.4 Extending the Commons

Once this basic separation of key repository has been established, along with the method for authorizing submissions and routing them to queries, it becomes possible to meaningfully entertain the question of including earlier stages. Especially with testing being scarce, confirmation occurs late and involves the contact tracing processes of public health authorities. Prior to that stage, we have Probable Covid (where a diagnosis has been performed based on defined symptom criteria), preceded by Suspected Covid, and often preceded by individual state of concern. Associated with each stage is a process, an (increasingly large) set of principals who might authorize a submission, and a reduced significance in conveying exposure. For example, the physician or health service making a Probable Covid diagnosis (and typically also authorizing testing) would be the natural principal to authorize the individual to submit "Probable Keys". The privacy-sensitive protocol could access these, as well as the "Confirmed Keys" referred to as Diagnosis Keys in the EN protocol. The protocol permits the distinction to be reflected in metadata carried along with the keys. Thus they might factor in, perhaps with lesser weight, into the risk score.

E Conclusion

With these two simple innovations, we can move past the conflicts and controversy and on to utilizing privacy-sensitive mobile contact tracing to improve the efficacy of public health measures - thereby saving lives and unnecessary suffering - while respecting civil liberty. The companies should provide the interoperable building blocks without themselves getting into the business of providing the Apps or holding the data. They can maintain their privacy-first, decentralization posture, but should advocate for an interoperable COVID key Commons, rather than dictating policy on the app ecosystem. Government actors can regain policy determination and influence the app ecosystem so as to best tailor offerings to their constituents. The tailoring could include questions around how best to provide a Commons of appropriate scale, and utilize physical measures to relate anonymous key reports to the places within their jurisdiction. Government actors already have extensive experience with civic infrastructure such as security cameras, traffic signals and health inspections. They can apply the similar principles towards recognizing the importance of public awareness, potential for creating stigma, and bringing benefit to non-participating members of the communities. The technology can assist the contact tracing process, but should not be dictating it or replacing the human relationship of the patient and the health professionals performing interviews and care. The trust that is built there is what makes opt-in approaches viable, and only with appropriate individual protections.

References

[1] A. Inc., *"Apple and Google partner on COVID-19 contact tracing technology"*, 10 April 2020 (accessed 25 May 2020). [Online]. Available: https://web.archive.org/web/20200410171128/https://www.apple.com/newsroom/2020/04/apple-and-google-partner-on-covid-19-contact-tracing-technology/

[2] J. Chan, D. Foster, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, P. Sharma, S. Singanamalla, J. Sunshine, and S. Tessaro, "Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing," 2020.

[3] C. e. a. Troncoso, *Decentralized Privacy-Preserving Proximity Tracing*. [Online]. Available: https://github.com/DP-3T/documents/blob/master/DP3T%20White%20Paper.pdf

[4] *20 April 2020 - One Month On*, 20 April 2020 (Accessed 24 May 2020). [Online]. Available: https://web.archive.org/web/20200524045214/https://support.tracetogether.gov.sg/hc/en-sg/articles/360046475654-20-April-2020-One-Month-On

[5] J. Lu, J. Gu, K. Li, C. Xu, W. Su, and Z. e. a. Lai, "Covid-19 outbreak associated with air conditioning in restaurant, guangzhou, china, 2020," *Emerging Infectious Disease*, 2020, https://doi.org/10.3201/eid2607.200764.

[6] E. Y.-J. Kang, *Asian Americans Feel The Bite Of Prejudice During The COVID-19 Pandemic*, National Public Radio, 31 March 2020 (Accessed 2 April 2020). [Online]. Available: https://web.archive.org/web/20200402203217/https://www.npr.org/local/309/2020/03/31/824397216/asian-americans-feel-the-bite-of-prejudice-during-the-c-o-v-i-d-19-pandemic

[7] H. Shin and J. Smith, "South korea scrambles to contain nightclub coronavirus outbreak," *Reuters*, 2020. [Online]. Available: https://www.reuters.com/article/us-health-coronavirus-southkorea/south-korea-scrambles-to-contain-nightclub-coronavirus-outbreak-idUSKBN22N0DA

[8] R. Hinch, W. Probert, A. Nurtay, M. Kendall, C. Wymant, M. Hall, and C. Fraser, "Effective configurations of a digital contact tracing app: A report to nhsx," 2020. [Online]. Available: https://github.com/BDI-pathogens/covid-19_instant_tracing/blob/master/Report-EffectiveConfigurationsofaDigitalContactTracingApp.pdf

[9] *Exposure Notification API launches to support public health agencies*, Apple Incorporated, Google LLC, 20 May 2020 (Accessed 24 May 2020). [Online]. Available: http://web.archive.org/web/20200524152945/https://blog.google/inside-google/company-announcements/apple-google-exposure-notification-api-launches/

[10] C. Ho, "Bay area coronavirus tests: Where can i get one?" 21 May 2020 (Accessed 25 May 2020). [Online]. Available: http://web.archive.org/web/20200525201614/https://www.sfchronicle.com/health/article/Where-can-I-get-a-coronavirus-test-in-the-Bay-15136054.php

[11] N. A. of County and C. H. Officials, "Directory of local health departments," 24 May 2020 (Accessed 24 May 2020). [Online]. Available: http://web.archive.org/web/20200524090932/https://www.naccho.org/membership/lhd-directory

[12] N. N. York, "What you need to know about New York's 'monumental' contact tracing program," 22 April 2020 (Accessed 24 May 2020). [Online]. Available: http://web.archive.org/web/20200524162632/https://www.nbcnewyork.com/news/coronavirus/what-you-need-to-know-about-new-yorks-monumental-contact-tracing-program/2385611/

T1]fontenc

Epione: Lightweight Contact Tracing with Strong Privacy

Ni Trieu[1], Kareem Shehata, Prateek Saxena[2], Reza Shokri[2], and Dawn Song[1,3]
[1]UC Berkeley, [2]National University of Singapore, [3]Oasis Labs

Abstract

Contact tracing is an essential tool in containing infectious diseases such as COVID-19. Many countries and research groups have launched or announced mobile apps to facilitate contact tracing by recording contacts between users with some privacy considerations. Most of the focus has been on using random tokens, which are exchanged during encounters and stored locally on users' phones. Prior systems allow users to search over released tokens in order to learn if they have recently been in the proximity of a user that has since been diagnosed with the disease. However, prior approaches do not provide end-to-end privacy in the collection and querying of tokens. In particular, these approaches are vulnerable to either linkage attacks by users using token metadata, linkage attacks by the server, or false reporting by users.

In this work, we introduce Epione, a lightweight system for contact tracing with strong privacy protections. Epione alerts users directly if any of their contacts have been diagnosed with the disease, protects the privacy of users' contacts from both central services and users, and provides protection against false reporting. As a key building block, we present a new cryptographic tool for secure two-party private set intersection cardinality (PSI-CA), which allows two parties, each holding a set of items, to learn the intersection size of their sets without revealing the intersection items. We specifically tailor it to the case of large-scale contact tracing where clients have small input sets and the server's database of tokens is much larger.

A Introduction

Contact tracing is an important method to curtail the spread of infectious diseases. The goal of contact tracing is to identify individuals that might have come into contact with a person that has been diagnosed with the disease, so they can be isolated and tested.

In the ongoing COVID-19 pandemic, contact tracing has been facilitated by mobile apps that detect nearby mobile phones using Bluetooth, and several countries / organizations have developed such apps. Such large-scale collection of personal contact information is a significant concern for privacy [1, 2, 3].

The main purpose of contact tracing applications—recording the fact that two or more individuals were near each other at a certain moment of time—seems to be at odds with their privacy. The app must record information about the individual's personal contacts and should be able to reveal this information (possibly, on demand) to some authorities.

Multiple ways have been proposed to protect user contact data, offering different privacy guarantees and coming at different implementation costs. For instance, in the recently released BlueTrace protocol used by the Singapore Government [5], users are guaranteed privacy from each other, but this model places complete trust in certain operating entities for protecting user information.

We consider a model where governments (or operators) do not store any such sensitive user information. Not only are such databases lucrative targets for cyber-attackers, in many jurisdictions the collection of such

information may raise public concerns or even conflict with privacy regulations. This is important not just for user privacy, but also because contact tracing is expected to be effective only when participation is high (e.g. 60% or more of the population [4]). Thus the overall success of the app could be limited if users are reluctant to use a contact tracing app due to privacy concerns.

In our model, the health authorities maintain a database of tokens corresponding to users which have been diagnosed with the disease. The tracing app periodically checks an untrusted server to determine if the user is potentially at risk. This is done in such a way that the server cannot deduce any information about the user which is not implied by the desired functionality. The users also learn no information beyond whether they may have been exposed to the disease.

Our model can also be contrasted to several other decentralized mobile contact tracing system/protocols, which we analyze in the full version of the paper [7]. As we see through our analysis, existing proposals or launched systems are vulnerable to one or more of the following privacy attacks:

(1) *Infection status / exposure source by users*: If tokens of users diagnosed positive are publicly released, Alice can determine which publicly-posted tokens match the log on her phone. This could reveal the time, for example, when Alice and the user diagnosed positive with the disease (Bob) were in close proximity, enabling her to identify Bob. Such identification is undesirable as people have been reported to harass individuals suspected to be the source of exposure to the disease [8], leading to the so-called "vigilante" problem.

(2) *Infection status by server*: If the server can determine which users have been diagnosed with the disease, this leaks the infection status of users to the server operator. This may not be a concern in jurisdictions where the server is operated by the health authority which already knows this information. However, in jurisdictions where the server is operated by another party that does not or should not have this information, this form of linkage can be a serious privacy threat.

(3) *Social graph exposure and user tracking*: If a central database is used to collect both sent and received tokens as in [9], or it is possible to infer the source of a sent token as in the case of [10], then the operator of this server is able to deduce all of the social connections of a user that is reported positive for the disease, including when and for how long each contact was made. This co-location information can also exacerbate the risk of users' location tracking [4, 3].

(4) *False-positive claims by users*: A user may claim to have been diagnosed with the disease when in reality, they are not. This would spread false information and panic other users, and reduce trust in the system.

Table 1 provides a brief comparison of different contact tracing systems with respect to security/privacy properties, required computational infrastructure and client's communication cost, all of which are important for a wide-scale real-world contact tracing. Details of the systems compared is discussed in the full version [7].

Our Contributions. In this work, we introduce Epione, a new system for decentralized contact tracing with stronger privacy protections than the existing systems. As a key primitive enabling Epione, we introduce a new private set intersection cardinality or PSI-CA, which is used to check how many tokens held by a user (client) match the tokens in a set stored on a server, without the user revealing their tokens. More formally, PSI-CA allows two parties, each holding a private set of tokens, to learn the size of the intersection between their sets without revealing any additional information. Our PSI-CA primitive is designed to be efficient for a large server-side database and a small client-side database, as is the case for contact tracing applications. Our new PSI-CA construction allows us to meet our privacy goals. With several other optimizations in our system design, we show that PSI-CA can make privacy-preserving contact tracing *practical*.

System	System Req.		Privacy Protection Against				Client
	Trusted Server	#	Infection Status		Social	False-positive	Comm.
			By User	By Server	Graph	User	Cost
TraceTogether [5]	Yes	1	Yes	No	No	Yes	$O(n)$
Baseline*	**No**	1	No	No	Most	Some	$O(N)$
Private Messaging [2]		3	No	Yes	Yes	No	
Epione		2	**Yes**	**Yes**	**Yes**	**Yes**	$O(n\log(N))$

Table 1: Comparison of contact tracing systems with respect to security, privacy, required computational infrastructure, and client communication cost. **Baseline** systems include Private Kit[13], Covid-watch [9], CEN [14], DP-3 [15], and PACT's baseline system [12]. Some of these systems provide a limited level of false-positive claim protection with an additional server (or healthcare provider), and most provide protection from social graph discovery. N is the total number of contact tokens from users diagnosed positive with the disease, n is the number of contact tokens recorded by an average user that need to be checked for disease exposure (Note that $\frac{N}{n}$ is typically the number of new positive diagnoses per day, thus $n << N$).

A.1 System Overview

Figure 1 shows an overview of the Epione system. Users of Epione want to be notified if any of the people they have been in contact with are later diagnosed with the disease. They do *not* want to reveal to other users their identity, reveal whether they have been diagnosed positive, be tracked over time, or reveal their contacts to any other organization.

We use a short-range network (such as Bluetooth) to detect when two users are within close range and exchange a randomly generated "contact token". All of the sent and received contact tokens are stored securely on the user's phone in the "sent token list" and "received token list", respectively. The received token list never leaves the user's phone in a form that can be used by anyone else, and the sent token list is only revealed to a healthcare provider on a positive diagnosis and with the user's consent. In Section E, we explain in detail how to generate and store the tokens.

In Epione, we assume that there is an untrusted service provider, which we call the Epione Server, which can collect the transmitted contact tokens from all users tested positive with the disease. The Epione server allows users to check whether they have received a token from a user who has since been diagnosed with the disease, without revealing to the server their tokens (and thus their contacts) and without the server revealing any information to the user about the tokens of users diagnosed positive beyond the count of contact tokens in common. We use secure computation techniques, particularly PSI-CA, for private matching. This prevents the Epione server from inferring linkages between users, as well as preventing users from inferring the diagnosis status of other users, or the source of any exposure to the disease.

It is assumed that a healthcare provider (such as a hospital) is aware of the identity of the user whom it diagnoses as having the disease. Thus, exposing the identity of the user diagnosed positive to the provider is not considered as a threat. It is also assumed that the healthcare provider keeps a local database of positively diagnosed users to be able to verify if a user was legitimately diagnosed positive. The healthcare provider collects (with the user's consent) the list of "sent tokens" from a positively diagnosed user's app and sends it to the Epione server, which the latter adds to a database of contact tokens from such users.

Note that in this model the server does not know the identity of the user diagnosed positive. It is not hard to imagine collusion between the healthcare database and the backend server for Epione, say by a state actor or attacker within the healthcare provider. Even then, the sent tokens are not useful for identifying any contacts or any other private information. Since tokens are randomly generated, the attacker would need to know which users received those tokens to re-identify them. Section E shows that the Epione server never learns the received tokens of any user and thus linkage is not possible.

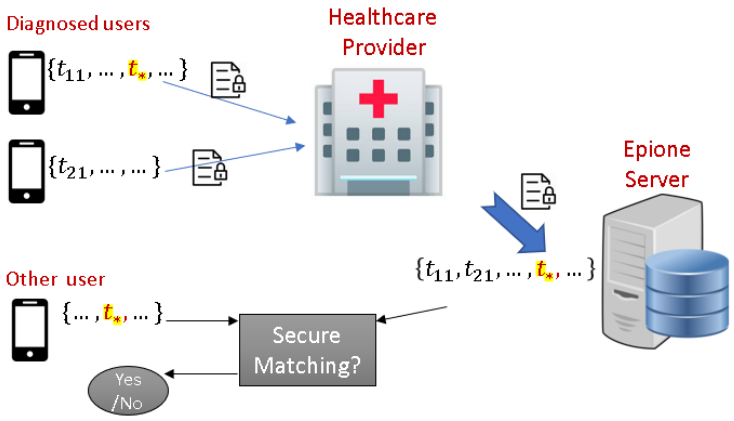


Figure 1: Overview of our Epione system. When a person is diagnosed with the disease, the healthcare provider collects their sent tokens and forwards them to the Epione server encrypted under the server's public key (actually, the PRG seed is collected to reduce communication costs). The Epione server decrypts the received ciphertexts from the healthcare provider and obtains the transmitted tokens of all patients diagnosed positive. Next, each user's app uses PSI-CA to compare their set of received tokens with the set of sent tokens stored on the Epione server. If the intersection size is more than zero, then the user is alerted that they may have been exposed to the disease.

B Related Work

We begin by discussing previous approaches to contact tracing, and then current approaches to secure computation and private set intersection, which form the basis for our own PSI-CA used in Epione.

B.1 Contact Tracing Approaches

Due to the rapid spread of the COVID-19 pandemic and the importance of contact tracing, many research groups have been developing tools to improve contact tracing. Most schemes either (1) rely on and expose data to a trusted third-party, such as TraceTogether [5], or (2) uses a decentralized/public list approach such as COVID-Watch [9], PACT [12], or Google/Apple [11] that allows users to infer linkages such as exposure sources.

In this work, we focus on the latter approach. Covid-watch [9], Private Kit [13][1], PACT's baseline design [12], and Google/Apple [11] are all variations on this design. Some use pseudo-random number generators, and upload seeds for the sent token lists to reduce communication and storage costs at the expense of greater cost for comparisons. All of these designs are susceptible to linkage attack by either users, the server, or both. Some offer protection against false-positive claims. We refer the reader to the full verion of this paper [7] for additional discussion of decentralized contact tracing.

In some of the above systems, each phone has to compare the publicly posted contact tokens against their own history, which requires to them download all public tokens. This requires significant bandwidth and places a burden on mobile devices.

[1]PrivateKit claims that in V3 they will introduce strong privacy protections, but as of writing this paper the protocol to do so has not been announced.

B.2 Secure Computation and Private Set Intersection

Private set intersection (PSI) refers to a cryptographic protocol that allows two parties holding private datasets to compute the intersection of these sets without either party learning any additional information about the other's dataset. PSI has been motivated by many privacy-sensitive real-world applications. It does, however, reveal the intersection elements to at least one party. In many scenarios (such as contact tracing) it is preferable to compute some function of the intersection without revealing the elements in it, such as whether intersection size is more than a given threshold. Limited work has focused on this so-called f-PSI problem. The Diffie-Hellman homomorphic encryption approach in [16] is preferable in many real-world f-PSI applications[2], due to its more reasonable communication complexity.

C Problem Statement and Security Goal

C.1 Problem Definition

We define the problem of contact tracing based on token exchange as follows. Various clients communicate with each other and with a contact tracing service. The service is provided by one or more servers. The overall system consists of the following procedures:

- $\texttt{Generate}(\kappa) \rightarrow t$: Client uses the `Generate` function to generate contact tokens, t, to be exchanged with other users. The function takes a security parameter κ as input.
- $\texttt{Exchange}(t_a) \rightarrow t_b$: The client (client A) uses the `Exchange` function to exchange tokens with another user (client B). Client A sends token t_a to B, and receives t_b from client B. Client A then stores t_b in the "received tokens list" ($\mathbf{T}_R$), and client B stores t_a in their "received tokens list".
- $\texttt{Query}(T_R, S) \rightarrow a$: With a set $\mathbf{T}_R$ of received tokens from `Exchange`, the client uses the `Query` function to query the server S and get an answer indicating how many of their tokens came from users currently diagnosed positive for the disease.

C.2 Security Goal

We consider a set of parties who have agreed upon a single function f to compute (such as contact tracing) and have also consented to give f's final result to some particular party. At the end of the computation, nothing is revealed by the computational process except the final output. In real-world execution, the parties often execute the protocol in the presence of an adversary $\mathcal{A}$ who corrupts a subset of the parties. In the ideal execution, the parties interact with a trusted party that evaluates the function f in the presence of a simulator Sim that corrupts the same subset of parties.

For simplicity, we assume there is an authenticated secure channel between each pair of clients, and client-server pair (e.g., with TLS). In this work, we consider a model with non-colluding servers. A desirable contract tracing system would make an honest user's actions perfectly indistinguishable from actions of all other honest users as well as servers. Thus, an ideal security system property would guarantee that executing the system in the real model is equivalent to executing this system in an ideal model with a trusted party. In particular, Epione provably provides all of the functions of contact tracing while protecting against the linkage attacks and false-positive claims described in Section A.

[2]Google Security Blog, June 19, 2019 "Helping organizations do more without collecting more data"

D Preliminaries

In this work, the computational and statistical security parameters are denoted by κ, λ, respectively. For $n \in \mathbb{N}$, we write $[n]$ to denote the set of integers $\{1, \ldots, n\}$.

Definition 1: [17] Let $\mathcal{G}(\kappa)$ be a group family parameterized by security parameter λ. For every probabilistic adversary $\mathcal{A}$ run in polynomial time in λ, we define the advantage of $\mathcal{A}$ to be $|\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(g, g^a, g^b, g^c) = 1]|$, where the probability is over a random choice G from $\mathcal{G}(\lambda)$, random generator g of G , random $a, b, c \in [|G|]$ and the randomness of $\mathcal{A}$. We say that the Decisional Diffie–Hellman assumption holds for G if for every such $\mathcal{A}$, there exists a negligible function ϵ such that the advantage of $\mathcal{A}$ is bounded by $\epsilon(\lambda)$.

Definition 2: [18] A pseudorandom number generator (PRG) is a function that, once initialized with some random value (called the seed), outputs a sequence that appears random, in the sense that an observer who does not know the value of the seed cannot distinguish the output from that of a (true) random bit generator.

Private Information Retrieval. Private Information Retrieval (PIR) allows a client to query information from one or multiple servers in a such way that the servers do not know which information the client requested. Recent PIR [19, 20, 21] reduces communication cost to logarithmic in the database size. In PIR, the server(s) hold a database DB of N strings, and the client wishes to read item $DB[i]$ without revealing i.

Chor, et al. [22] define a variant of PIR called keyword PIR, in which the client has an item x, the server has a set S, and the client learns whether $x \in S$. In this paper, we are interested in Keyword PIR based on both 1-server PIR [20, 21] and 2-server PIR [23, 24] with different trade-offs.

Private Set Intersection Cardinality. Private set intersection cardinality (PSI-CA) is a two-party protocol that allows one party to learn the intersection size of their private sets without revealing any additional information.

E Our Epione System

We now present the Epione system in detail, the construction of which closely follows the high-level overview presented in Section A.1. Recall that Epione aims to alert any users who have, within the infection window, come into contact with another user who has been diagnosed positive with an infectious disease.

Epione's design combines several different cryptographic primitives. We refer the reader to Section D and Section F for more details on the cryptographic gadgets used here. The Epione system consists of four phases as follows.

Agreement and Setup Phase. The Epione server takes a security parameter λ as input, outputs a public-private key pair (pk, sk), and shares the public key with every user. Each user/client u_i generates a random PRG seed s_i which it uses to generate contact tokens in the next phase. As long as the server's configuration does not change, this phase does not need to be re-run. Whenever a new user registers with Epione, they only need to generate their own PRG seed, and the server shares the public key with the new user.

Token Generation. Similar to most recent contract tracing systems [2, 9, 14, 12], we use Bluetooth to exchange contact tokens whenever two users are in close proximity. The $\texttt{Generate}(s_u, d_i, j) \to t_{u,i,j}$ function is used to generate tokens of κ bits each to be sent by user u on day i and timeslot j. The precise details of the token generation are left as an implementation detail, so long as the following criteria are met:

- Tokens are indistinguishable from random by anyone not in possession of the user's seed s_u. In other words, the `Generate` function acts as a PRG as defined in Section D.

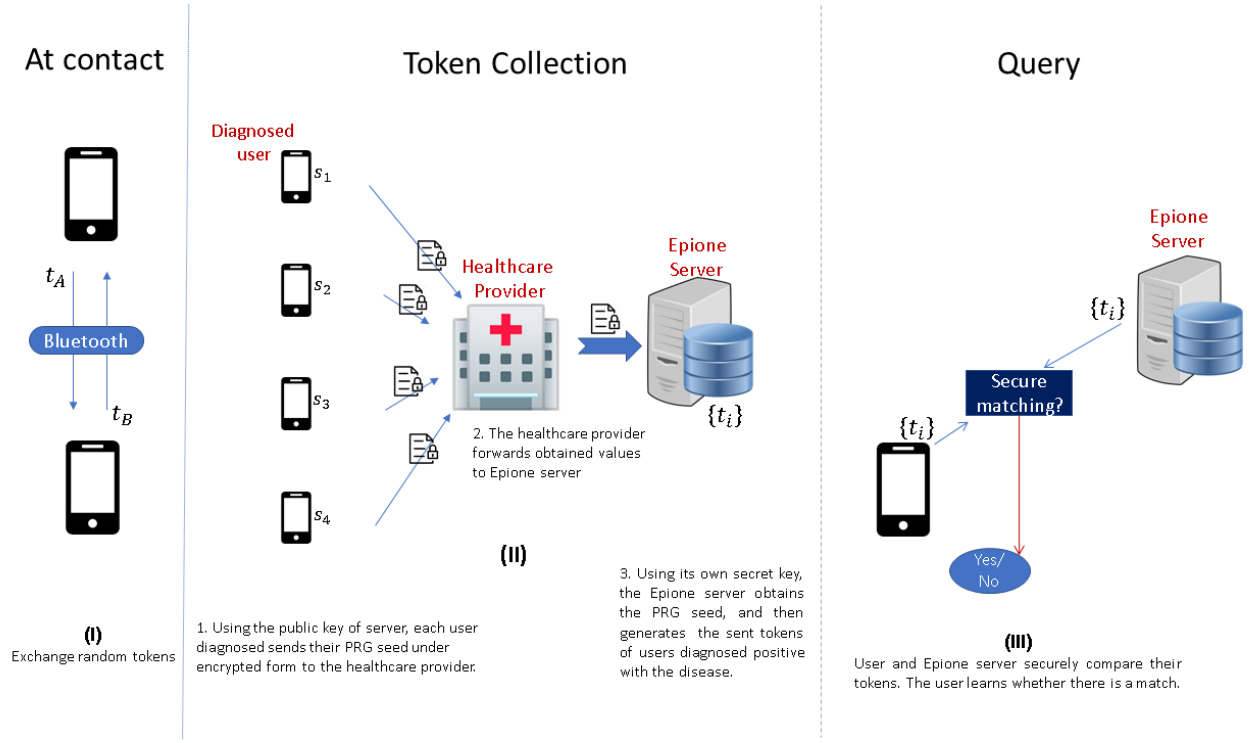


Figure 2: Epione System Design without Agreement and Setup Phase. (I) Tokens are exchanged when two users are in close proximity. (II) When a user is diagnosed with the disease, the user encrypts their PRG seed using the public key of the Epione server, and gives the encrypted value to the healthcare provider, who then transmits it to the Epione server. Using its private key, the Epione server decrypts the received ciphertexts and obtains the PRG seeds of diagnosed users. The Epione server generates the sent tokens of users diagnosed positive using the PRG. (III) Each user invokes a secure matching algorithm with the Epione server, where the user's input is their received tokens and the server's input is the database of tokens from users diagnosed with the disease. The user learns only whether (or how many) tokens there are in common between the two sets, while the Epione server learns nothing.

- Tokens can be deterministically generated for the given day d_i, and time slot j using a secret seed, s_u, such that when a user gives their seed to the Epione server, the server is able to regenerate the tokens sent by the user.
- All users and the Epione server agree on the method used to generate tokens, the time intervals, and day numbering.

Contact. As illustrated in Figure 2 part I, when two users, say Alice and Bob, enter within close proximity, Epione detects this condition with a short range network such as Bluetooth, and then uses that network to exchange tokens using the function `Exchange`. Alice generates token $t_a \leftarrow \texttt{Generate}(s_a, d_i, j)$, where s_a is Alice's private seed, d_i is the current day, and j is the current time slot. Similarly Bob generates token $t_b \leftarrow \texttt{Generate}(s_b, d_i, j)$. Alice sends t_a to Bob, and Bob send t_b to Alice.

Alice then adds the token received from Bob, t_b, to her set of received tokens, $\mathbf{T}_{R,A}$, and Bob adds t_a to $\mathbf{T}_{R,B}$. We use $\mathbf{T}_{S,A}$ to represent the set of tokens Alice has sent to other users (which includes t_a), though Alice does not actually store such a list since it can be regenerated at any time from her private seed. Alice and Bob discard received tokens that are older than the infection window (e.g. 14 days for COVID-19).

Positive Diagnosis and Token Collection. When a user (u_i in general) is diagnosed with the disease, the user encrypts their PRG seed using the public key of the Epione server and gives that to the healthcare provider (provided the user consents to this, of course). The healthcare provider gathers the seeds from several users diagnosed positive, shuffles them, and transmits the set of seeds over a secure channel to the Epione server. Using its private key, the Epione server decrypts the received values to obtain the secret PRG seeds. The Epione server can then generate all of the tokens for the infection window sent by users diagnosed positive with the disease, $\widehat{\mathbf{T}}_S$. The token collection process is shown in part II of Figure 2.

Two servers are used at this phase to prevent any one server from knowing both the diagnosis status of a user and their sent tokens. This is useful in the case that the Epione server is operated by an untrusted party, such as a commercial provider, that should not have access to sensitive information such as a user's diagnosis. If such protection is not needed, for example if the Epione server is operated by a health authority that already has access to the infection status of users and can be trusted not to try to discern a user's diagnosis status from the token collection process, then both services can be provided by the same server.

Alternatively, the healthcare provider could provide a token to the user that the user then provides the Epione server when they upload their tokens to prove that they have a legitimate positive diagnosis. This would allow the Epione server to verify that the user's claim is legitimate, but does not protect the user from the server linking them to a positive diagnosis.

Query. Recall from the contact phase that each user u_i keeps a list of tokens received from other users they have been in contact with within the infection window, $\mathbf{T}_{R,u_i}$. The query phase aims to securely compare the user's received contact tokens $\mathbf{T}_{R,u_i}$ with the Epione server's set of tokens sent by users diagnosed positive with the disease, $\widehat{\mathbf{T}}_S$. If there are any tokens in common, then user u_i has come into contact with an individual diagnosed positive within the infection window, and should be notified that they are at risk of having contracted the disease. This process is illustrated in part III of Figure 2.

The comparison of tokens is done by calling the `Query` function, which we implement using PSI-CA. We describe PSI-CA in detail in Section F. Note that revealing the intersection size is acceptable in the contact tracing application we consider, however, it is possible hide the intersection size as we discuss in the full version [7].

F Cryptographic Gadgets

This section provides more detail on the cryptographic tools we use to implement Epione, with a specific emphasis on our PSI-CA design and PIR. Extension to those tools is discussed in the full version [7].

F.1 PSI cardinality (PSI-CA) for asymmetric set sizes

F.1.1 Our technique

We start with a private set intersection (PSI) in the semi-honest setting, where two parties want to learn the intersection of their private set, and nothing else. The earliest protocols for PSI were based on the Diffie-Hellman (DH) assumption in cyclic groups. Currently, DH-based PSI protocols [25] are still preferable in many real-world applications due to their low communication cost.

DH-based PSI. Assume that the server has input $X = \{x_1, \ldots, x_N\}$ and client has input $Y = \{y_1, \ldots, y_n\}$. Given a random oracle $H : \{0,1\}^* \to G$, and a cyclic group G in which the DDH assumption holds, the basic DH-based PSI protocol is shown in Figure 3. Intuitively, the client sends $\{H(y_i)^r\}_{y_i \in Y}$ for some random, secret exponent r. The server raises each of these values to the k power, and the client can then raise these results to the $1/r$ power to obtain $\{H(y_i)^k\}_{y_i \in Y}$ as desired.

PARAMETERS: cyclic group G of order p; random oracle H.
INPUTS: Server has input $X = \{x_1, \ldots, x_N\}$; client has input $Y = \{y_1, \ldots, y_n\}$.
PROTOCOL:

1. Client chooses $r \leftarrow \mathbb{Z}_p$, for each $y_i \in Y$ sends $m_i = H(y_i)^r$ to the server.
2. Server chooses $k \leftarrow \mathbb{Z}_p$ and for each $i \in [n]$, sends $m'_i = m_i^k$ to the receiver in a randomly permuted order.
3. For each $i \in [n]$, the client computes $v_i = (m'_i)^{1/r}$.
4. For each $x_i \in X$, the server computes $u_i = H(x_i)^k$ and sends $U = \{u_i \mid x_i \in X\}$ (randomly permuted) to the client.
5. [PSI -CA output] The client outputs $|\{i \in [n] \mid v_i \in U\}|$.

Figure 3: DH-based PSI protocol and extension to PSI-CA with changes highlighted.

From DH-based PSI to PSI cardinality (PSI-CA). If the client uses the same r for every item, it is possible to extend the basic PSI algorithm to compute functions such as intersection set size (cardinality) without revealing the intersection items by having the server shuffle the items. This observation was suggested by [25] and recently was incorporated into private intersection sum [16], which allows two parties to compute the sum of payloads associated with the intersection set of two private datasets, without revealing any additional information. Clearly, PSI-CA is a special case of private intersection sum, where the payload is constant and equal to 1.

Figure 3 also shows the extension to PSI-CA with the highlighted changes. The key idea to transform PSI into PSI-CA is that instead of sending m'_i in step 2 of Figure 3 in order, the server shuffles the set in a randomly permuted order. Shuffling means the client can count how many items are in the intersection (PSI-CA) by checking whether $v_i \in U$, but learns nothing about which specific item was in common (e.g. which v_i corresponds to the item y_j). Thus, the intersection set is not revealed.

From PSI-CA to PSI-CA for asymmetric sets. In many applications, including contact tracing, the two parties (client and server) have sets of extremely different sizes. A typical client has less than 500 new tokens per day, while the server may have millions of tokens in its input set. In PSI, most work is optimized for the case where two parties have sets of similar size, and as such their communication and computation costs scale with the size of the larger set. For contact tracing, it is crucial that the client's effort (especially communication cost) be sub-linear in the server's set size. More practically, we aim for communication of at most a few megabytes in a setting where the client is a mobile device.

We observe that the last two steps of Figure 3 are similar to the function performed by keyword PIR, which is communication-efficient in the conventional client-server setting. Keyword PIR allows clients to check whether their item is contained in a set held by a server, without revealing the actual item to the server. Therefore, step 4 and 5 of Figure 3 can be replaced by keyword PIR. Concretely, after step 3, the client has an input set $V = \{v_1, \ldots, v_n\}$ and the server has input set $U = \{u_1, \ldots, u_N\}$. The client sends a multi-query keyword PIR request with all of the elements in V to be queried against U on the server. From the PIR response, the client can count the number of $v_i \in U$ to find the set size, without revealing to the server the actual values in V and without the client learning any more information about U.

F.1.2 Protocol

Our semi-honest PSI-CA protocol is presented in Figure 4, following closely the description in the previous subsection. The client runs keyword PIR searches for each $v_{i\in[n]}$ in a set U held by the server. For communication and computation efficiency, the values of both u_i and v_i can be truncated, and the protocol is still correct as long as there are no spurious collisions. We can limit the probability of such a collision to $2^{-\lambda}$ by truncating to length $\lambda + log(N)$ bits. In Figure 4, we use a truncation function $\tau(z)$ which takes z as input and returns the most significant $\lambda + log(N)$ bits of z.

PARAMETERS: cyclic group G of order p; random oracle H, Multi-query Keyword PIR primitive (Section D), a truncate function $\tau(z)$ takes z as input and returns first $\lambda + log(N)$ bits of z.
INPUTS: Server 1 has input $X = \{x_1, \ldots, x_N\}$; client has input $Y = \{y_1, \ldots, y_n\}$; Server 2 has no input
PROTOCOL:

1. Server 1 chooses $k \leftarrow \mathbb{Z}_p$, and computes dataset $U = \{\tau\big(H(x_i)^k\big) \mid i \in [N]\}$. Server 1 sends U to Server 2
2. Client chooses $r \leftarrow \mathbb{Z}_p$, and sends $m_i = H(y_i)^r$ for each $y_i \in Y$ to the server 1.
3. Server 1 chooses a random permutation $\pi : [n] \rightarrow [n]$. For each $i \in [n]$, sends $m'_i = (m_{\pi(i)})^k$ to the client.
4. For each $i \in [n]$, the client computes $v_i = \tau\big((m'_i)^{1/r}\big)$.
5. Parties invoke Multi-query Keyword-PIR:
   - Server 1 acts as Keyword-PIR server 1 with dataset U
   - Server 2 acts as Keyword-PIR server 2 with dataset U
   - Client acts as Keyword-PIR client with $V = \{v_i \mid i \in [n]\}$
   - Client learns whether $v_i \in U, \forall i \in [n]$
6. Client outputs $|V \cap U|$

Figure 4: Our semi-honest PSI-CA protocol for asymmetric sets, and extension to 2-server PIR based PSI-CA with changes highlighted

PSI-CA Cost. The server and client must communicate (1) $O(n)$ group elements, (2) $O(n)$ homomorphically encrypted selection vectors for Keyword-PIR. If Keyword-PIR uses $O(\log(N))$ bits for each vector[3], the total communication cost is $O(n \log(N))$ bits. We provide more analysis of performance in Section G and the full version of the paper [7]. The client's computation is $O(n)$ and the server's computation is $O(nN)$.

The two-server PIR model can be used to speed up the server side computation by avoiding homomorphic encryption operations.

[3]There is a tradeoff between communication and computation complexity in PIR/Keyword-PIR as discussed in [21]. Traditional PIR is $O(\log(N))$ or $O(polylog(N))$ for query vectors, but some schemes trade slightly higher communication complexity for reduced computational complexity.

F.2 PSI-CA with 2-server PIR

Recall that the client and server invoke Keyword PIR in step 5 of Figure 4. To speed up the computational overhead on the server side, we introduce a second, independently operated server. The primary server sends the dataset U to the second server after it has been computed. By DDH, the second server learns nothing about the item x_i from u_i.

The client sends PIR queries with keyword v_i to both servers, and learns whether $v_i \in U$ and nothing else. Neither PIR server learn anything about the client's query as long as the two servers do not collude.

With 2-server PIR, the computation cost of PIR contains only symmetric-key operations, using approximately $2N$ PRF calls, and the communication cost of PIR is $O(log(N))$ bits. The highlights in Figure 4 shows the changes in PSI-CA to go from single-server to 2-server PIR.

G Implementation Choices and Performance Estimates

The main computation cost of our solution is PSI-CA algorithm, which itself is dominated by (1) token transforms (exponentiation) and (2) keyword PIR [20, 21]. We first propose the parameters of our estimation in the following subsection, then summarize the overall system performance. We refer reader to the full verion of the paper [7] for additional analysis of the Epione's performance.

G.1 Parameters and token storage

Assuming that a contact token is generated every 15 minutes for approximately 20 hours a day, then each user sends 80 distinct 128-bit tokens per day. If we assume that a user also receives approximately the same number of tokens and the infectious period is 14 days for COVID-19, then each client receives a total of $n = 1120$ over 14 days. If there are 5,000 new cases per day, the server receives $N = 1120 \times 5000 = 5.6 \times 10^6$ new tokens per day.

In Epione, the server maintains a list of tokens from positive patients for the duration of the infectious window. When a user is diagnosed positive for the disease, they give all of their sent contact tokens for the infection window (or the seeds to generate them) to the server. Rather than storing these by day they were exchanged, it is both more efficient and improves privacy for the server to store them by the day the server received the tokens. This way clients can query only for new tokens that have arrived since they last checked, rather than querying against the entire set.

If there are 5,000 new cases per day, the server receives 5.6×10^6 new tokens per day. Storing both sent and received tokens requires 35 KiB of storage on the client (this can be reduced to 18 KiB if sent tokens are generated with a PRG and only the seed needs to be stored). Assuming the server needs to keep 15 days of tokens in case clients are offline, the total storage for tokens is 1.25 GiB.

G.2 Implementation optimization: Database shape

The bottleneck for scaling PSI-CA to serve a large dataset to a large number of users is PIR. In order to scale up PIR, we propose using a bucket system similar to the password checkup design in [21]. First, the database is split into n_{shards} shards (sometimes referred to as megabuckets). Transformed tokens are grouped into buckets, each bucket holding the same number of tokens, with dummies added as needed. Rather than performing keyword PIR, normal PIR with a bucket address is used. Since tokens are expected to have a uniform distribution (both before and after transformation), tokens should be uniformly distributed across shards and buckets. As such, the bucket addresses can simply be the first $\log_2(n_{shards} n_{buckets})$ most significant bits of the transformed token itself. Alternatively, a fast hash of the transformed token into the number of bits needed can be used. Recall that each transformed token is truncated to 74 bits before being stored in the database. We use the top bits of the token to be the shard index and bucket address, and only store the remaining bits.

For example, if there are 5.6 million tokens in the server's set, the database can be sharded into 8 sets each with approximately 700,000 tokens (again, assuming a uniform distribution of tokens). If each shard holds 2^{18} buckets, then each bucket holds $\lceil \frac{700{,}000}{2^{18}} \rceil = 3$ transformed tokens, with dummies added as necessary to pad buckets to the required length. The first three most significant bits of the transformed token are used as the shard number, and the following 18 bits of the transformed token are then the bucket address. Since each transformed token is stored as $74 - 3 - 18 = 53$ bits, each bucket has 20 bytes. More detail of the implementation is present in the full version [7].

G.3 Overall PSI-CA Performance Estimates

A major advantage of the Epione design is that the database shape described in the previous section can be tuned to fit the needs of the application and adjusted over time. If on a given day there is a spike in the number of tokens from users diagnosed with the disease, the number of database shards can be increased or the size of the buckets increased.

Using the parameters in Section G.1 as a starting point and a few assumptions on the database shape, we estimate that single-server PIR-based PSI-CA will take approximately 35 seconds to complete a query. If the query is done in the background without the user waiting on a response, then the query can be done in the cloud as a lower-priority batch processing job, and server resources can be scaled up to meet the number of users required. This was an intentional tradeoff for network efficiency. If the server does some caching of the query keys, then only 37 MiB of network traffic is needed.

The 2-server approach reduces both server computational load and produces a large savings in network bandwidth, but requires an independent party and thus may increase infrastructure costs. Concretely, to complete a query, it requires 1.8 seconds and 679 KiB data transmitted.

We believe that the Epione solution proposed is feasible in practice. This will be studied further at implementation to determine the optimal configuration.

Acknowledgments

We thank Min Suk Kang, Ilya Sergey, Jun Han, Xiaoyuan Liu, Duong Hieu Phan, Jiaheng Zhang, Tiancheng Xie, and Lun Wang for helpful discussion. This material is in part based upon work supported by the National Science Foundation(NSF) under Grant No. TWC-1518899, DARPA under Grant No. N66001-15-C-4066, Center for Long-Term Cybersecurity (CLTC), and IC3 industry partners. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF, DARPA, CLTC or IC3.

References

[1] Center for disease control and prevention. Contact Tracing : Part of a Multipronged Approach to Fight the COVID-19 Pandemic. https://www.cdc.gov/coronavirus/2019-ncov/php/principles-contact-tracing.html

[2] H. Cho, D. Ippolito, and Y. W. Yu. Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs. *arXiv* 2003.11511 2020

[3] R. Shokri, G. Theodorakopoulos, J.Y. Le Boudec, and J.P. Hubaux. Quantifying location privacy. IEEE symposium on security and privacy 2011

[4] A.M. Olteanu, K. Huguenin, R. Shokri, M. Humbert, and J.P. Hubaux. Quantifying interdependent privacy risks with location data. IEEE Transactions on Mobile Computing 2016

[5] B. Jason, K. Joel, T. Alvin, S. H. Chai, Y. Lai, T. Janice, and T. A. Quy. Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders. `https://bluetrace.io/`

[6] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dorner, M. Parker, D. G. Bonsall, and C. Fraser. Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. *medRxiv*, 2020

[7] N. Trieu, K. Shehata, P. Saxena, R. Shokri, and D. Song . Epione: Lightweight contact tracing with strong privacy. *arXiv:2004.13293* 2020

[8] More scary than coronavirus: South korea's health alerts expose private lives. https://www.theguardian.com/world/2020/mar/06/ more-scary-than-coronavirus-south-koreas-health-alerts-expose-private-lives

[9] Covid-watch. https://www.covid-watch.org/.

[10] Tracetogether. https://www.tracetogether.gov.sg/.

[11] Apple and google partner on covid-19 contact tracing technology. `https://www.apple.com/newsroom/2020/04/apple-and-google-partner-on-covid-19-contact-tracing-technology/`

[12] J. Chan, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, S. Singanamalla, J. Sunshine, and S. Tessaro. Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing. 2020

[13] R. Raskar, I. Schunemann, R. Barbar, K. Vilcans, J. Gray, P. Vepakomma, S. Kapa, A. Nuzzo, R. Gupta, A. Berke, D. Greenwood, C. Keegan, S. Kanaparti, R. Beaudry, D. Stansbury, B. B. Arcila, R. Kanaparti, V. Pamplona, F. M. Benedetti, A. Clough, R. Das, K. Jain, K. Louisy, G. Nadeau, V. Pamplona, S. Penrod, Y. Rajaee, A. Singh, G. Storm, and J. Werner. Apps gone rogue: Maintaining personal privacy in an epidemic. 2020

[14] Cen. https://github.com/Co-Epi/CEN

[15] Dp-3t. https://github.com/DP-3T/documents

[16] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, M. Raykova, S. Saxena, K. Seth, D. Shanahan, and M. Yung. On deploying secure computing commercially: Private intersection-sum protocols and their business applications. *Cryptology ePrint Archive*, Report 2019/723, 2019. https://eprint.iacr.org/2019/ 723.

[17] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans.* Inf. Theor., September 2006

[18] F. Koeune. Pseudorandom Number Generator. *pp. 995–996. Boston, MA: Springer US*, 2011.

[19] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. in *ICALP* 2005 , vol. 3580 of LNCS, pp. 803–815, Springer, Heidelberg, July 2005

[20] S. Angel, H. Chen, K. Laine, and S. T. V. Setty. PIR with compressed queries and amortized query processing. in 2018 *IEEE Symposium on Security and Privacy*, pp. 962–979, IEEE Computer Society Press, May 2018.

[21] A. Ali, T. Lepoint, S. Patel, M. Raykova, P. Schoppmann, K. Seth, and K. Yeo. Communication– computation trade-offs in PIR. *Cryptology ePrint Archive*, Report 2019/1483, 2019. https:// eprint.iacr.org/2019/1483

[22] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. *Cryptology ePrint Archive*, Report 1998/003, 1998. http://eprint.iacr.org/1998/003

[23] E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. *EUROCRYPT* 2015. Springer, Heidelberg, April 2015.

[24] E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing: Improvements and extensions. *ACM CCS* 2016.

[25] B. A. Huberman, M. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. in *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pp. 78–86, ACM, 1999.

T1]fontenc colorlinks]hyperref

BeeTrace: A Unified Platform for Secure Contact Tracing that Breaks Data Silos

Xiaoyuan Liu[1,2], Ni Trieu[2], Evgenios M. Kornaropoulos[2], and Dawn Song[2,3]
[1]Shanghai Jiao Tong University, [2]UC Berkeley, [3]Oasis Labs

Abstract

Contact tracing is an important method to control the spread of an infectious disease such as COVID-19. However, existing contact tracing methods alone cannot provide sufficient coverage and do not successfully address privacy concerns of the participating entities. Current solutions do not utilize the huge volume of data stored in business databases and individual digital devices. This information is typically stored in data silos and cannot be used due to regulations in place. To successfully unlock the potential of contact tracing, we need to consider both data utilization from multiple sources *and the* privacy *of the participating parties. To this end, we propose BeeTrace, a unified platform that breaks data silos and deploys state-of-the-art cryptographic protocols to guarantee privacy goals.*

A Introduction

The lessons learnt from the recent events of the COVID-19 pandemic show the need to effectively track the chain of infection. This task entails locating exposed individuals as well as identifying places that infected individuals have contaminated. The act of *contact tracing via contact chain* can significantly contribute in controlling the outbreak of an infectious disease. Unfortunately, traditional manual contact tracing approaches are not sufficient [1] against viruses that spread quickly via multiple transmission routes. Digital contact tracing has the potential to be more effective by efficiently utilizing large amounts of data from multiple sources. In this work we focus on *digital contact tracing*, as opposed to manual.

Shortcomings of Current Approaches. Most existing contact tracing applications focus on monitoring direct contacts between individuals. While this is a significant step forward, such an approach does not consider the case of *indirect contact* where individuals appear at the same location *after* a contamination event. To address this phenomenon, we view contact tracing as a *continuous monitoring procedure* that identifies both individuals who may have been exposed to the virus as well as high-risk places.

The data sources used by existing contact tracing solutions are often limited, which in turn leads to coverage problems. Many contact tracing apps rely exclusively on GPS and/or Bluetooth for collecting contact information. Unfortunately this approach does not allow the significantly more challenging case of indirect contact tracing. Additionally, these sources are not always reliable since GPS suffers from location precision, while Bluetooth techniques exchange tokens only between devices that are *simultaneously at the same place* and as a result does not capture the case of indirect contact. The above state of affairs makes clear that we need *additional sources* to improve coverage, e.g., data from business records. Interestingly, manual contact tracing identified these needs

*nitrieu@berkeley.edu
†evgenios@berkeley.edu
‡dawnsong@berkeley.edu

and incorporates business data such as hotel and restaurant records to discover potential contact with carriers. However, in the digital world, business records are stored in data silos. An additional practical challenge is that records are often in different formats and are protected by strict legal regulations that forbid access to third-parties.

As a data collection and aggregation procedure, contact tracing needs to process *sensitive data* of user's location and interactions. Therefore, privacy needs to be a key characteristic of a contact tracing platform. In order for the platform to notify exposed individuals it has to collect personally identifiable information (PII), along with the contact history and daily trajectory. Exposure of this sensitive information in plaintext violates the privacy of users and exposes the participating parties to liability issues. To illustrate this point, an adversary that gets access to this information [2] can infer the daily habits of traced users as well as their social graph based on the contact history.

Our Contributions. In this work we propose BeeTrace, a unified contact tracing platform that breaks data silos and achieves efficient data utilization while protecting privacy. We apply two key designs: *unified data format with granularity adaptation* and *privacy-preserving distributed query framework*. Our contributions in this paper are the following: (1) We *define the goal* of contact tracing systems and provide a way to track *indirect contact* by specifying high-risk places as one of our tracing targets. (2) We put forth the setting of *multi-source data collection* in contact tracing. In this setup we include business-side participation. (3) We list *privacy requirements* and discuss information distribution options for the participating parties. (4) We incorporate all of the above insights in a unified contact tracing platform called BeeTrace. Our platform sets the vision for the standardization of the data format as well as the deployment of privacy-preserving distributed query processing. We provide use cases for a number of contact tracing scenarios and explain the corresponding cryptographic techniques, i.e., multi-party computation (MPC), and security protocols that meet our privacy requirements.

B Problem Definition

The high pre-symptomatic ratio of COVID-19 and the high basic reproduction number (R_0), i.e., the average number of infections caused by a single carrier, make contact tracing vital in addition to the isolation of symptomatic carriers [3]. Droplets, aerosol, contamination of surfaces, and fecal-oral contamination can all cause transmission [4]. As a result, once an infectious carrier contaminates a place, the risk of infection remains for a prolonged period of time.

Terminology. In this work we consider both *static places*, such as hotels and restaurants, as well as *dynamic places*, such as buses and taxis. Through medical tests we can verify the infection status of individuals. We use *diagnosed carriers* to refer to individuals who test positive and have not recovered yet. Asymptomatic, pre-symptomatic or mildly-symptomatic infectious individuals are often unaware of their infection before taking a medical test. We call these individuals *undiscovered carriers*. Since both diagnosed carriers and undiscovered carriers are infectious, we consider all the places they have visited as *high-risk places*. We describe the process of contact tracing using the following terminology, see Figure 1 for an illustration:

- **Tracing Starting Points.** This is a list of verified carriers as well as places for which a confirmed carrier visited. All entries of this list are accompanied by an official record that proves the authenticity of their contamination claim, e.g., for the case of a carrier a medical test result and for a case of a contaminated place a business transaction initiated by a confirmed carrier.
- **Tracing Targets.** This is a list of individuals and places that interacted with an entity from the list of tracing starting points. This list includes the following entities:
  - Individuals that have been within certain distance from a diagnosed carrier from the list of tracing starting points.
  - Individuals that have visited a contaminated place from the list of tracing starting points.

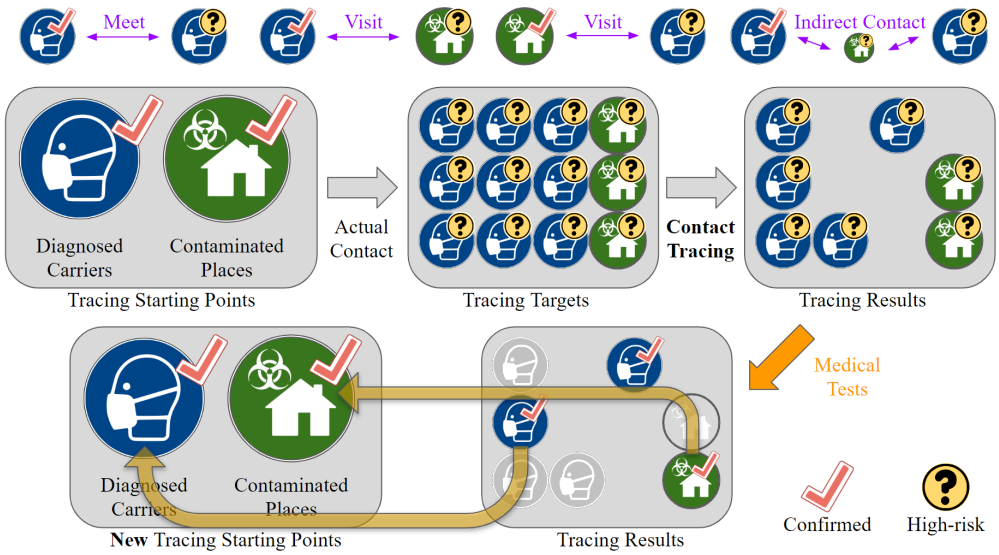


Figure 1: An illustration of the Contact Tracing process and its key terms. The top row presents potential direct and indirect contact scenarios. A contact tracing system needs to start from the confirmed cases defined as *tracing starting points* and compile the *tracing targets* that get a medical test to produce the *tracing results*. As a last step, newly diagnosed carriers and contaminated places are added to the tracing starting points.

  - Places that have been visited by a diagnosed carrier from the list of tracing starting points.

- **Contact Tracing.** This functionality takes as an input the tracing starting points, and outputs the the tracing targets. We use the term *tracing result* to refer to the output of the contact tracing functionality. A successful contact tracing should have a tracing result that matches precisely the true tracing targets.

A *contact tracing system* is an automated system that uses data from multiple sources to conduct contact tracing. Our vision for large-scale contact tracing involves (1) a *unified approach* of processing data from *multiple sources*, (2) and a focus on the *privacy requirements* for all participating individuals and businesses.

B.1 Contact Tracing from Multiple Sources: Necessity of Breaking Data Silos

A unified contact tracing system needs to break data silos and use multiple data sources to achieve high coverage. Figure 2 illustrates with a concrete example the necessity of breaking data silos. We note here that this is the running example that we use across our work and it is used for motivating our vision for BeeTrace.

In the following we list potential events from which a newly confirmed carrier, whom we call Alice, was infected either through direct or indirect contact by a known carrier, whom we call Bob. Then, given the events that took place during Alice's outing, see Figure 2, one of the following events took place: (1) Bob took the same taxi earlier that day, (2) Bob had lunch at that restaurant, (3) Bob walked at a very close distance to Alice at the park, (4) Bob was at the same subway car as Alice, (5) Bob stayed at the same hotel room.

Without records from multiple sources, e.g., transactions from the taxi company and the restaurant, it is highly unlikely that a contact tracing platform can identify which of the above events took place. As a result, users such as Alice would not have been notified and would probably spread the infection even further. To address the challenges of multi-source contact tracing we use *existing (traditional) data sources* and we create *new data sources* tailored to task of contact tracing.

Existing Data Sources. Notice that existing databases from businesses that Alice visited already record her contact history with both individuals and places. We envision a platform, namely BeeTrace, that can *facilitate a*

One sunny morning, Alice decides to visit her friend in a distant town. After taking a commercial flight to her friend's town, she calls a taxi using her mobile phone and goes to a famous restaurant for lunch with friends. Alice and her friend then go for a walk at a nearby park before parting ways. She takes the subway from the park to the airport but is informed that her flight was delayed. She changes her plans and decides to stay overnight at a nearby hotel. The next day,

- Alice receives a notification on her phone to self-isolate and seek a medical test.
- The hotel staff is also aware that Alice is at risk and calls to adviser her to self-isolate before taking a medical test.

Figure 2: Real-world example of contact tracing based on Alice's outing.

secure computation over multiple service providers such as ride-sharing companies, subway companies, restaurant businesses, and hotels, in order to perform accurate, scalable, and privacy-preserving contact tracing.

New Data Sources. As for new data sources that are focused on contact tracing data, one can either incorporate data from newly developed applications such as [5] or develop an independent contact tracing application. Most new applications use either Bluetooth or sensors such as mobile-phone camera for QR code scanning. In order to cover high-risk place tracing using Bluetooth, one potential approach is to install *Bluetooth beacons* in taxis, subways, and restaurants. These beacons transmit random identifiers that can be later found on a carrier's device. Such an approach is compatible with solutions like Google/Apple exposure notification systems [5]. This solution has a low implementation overhead if one is to follow the same format (Rolling Proximity Identifier) as the protocol of a deployed notification mechanism. Another approach is to put QR-codes in public places that can be scanned with a mobile phone camera and accurately locate the time and the location of a participant. This technique is successfully applied in China [6]. Both approaches can contribute to indirect contact tracing.

Proposed Approach. Interestingly, one can decouple the step of *collecting data* from the step of *aggregating data*. We only consider the case where the data that is collected by each participant is not transmitted to a centralized authority but rather stays in a thin client that is part of our platform. Specifically, our approach can integrate data from existing contact tracing apps into our aggregation platform for comprehensive coverage. Our platform can also integrate data from individual users' mobile phones and can be extended to consider other data sources including data sources collected by businesses. To achieve this functionality we propose the conversion of the data generated by third-party apps and we use them as a contact tracing input to our platform. A challenge that we had to overcome is that both the *precision* and the *format* of third-party generated data varies. A key-design of our platform is *a unified data format with granularity adaptation*, which is discussed in Section D.

B.2 Privacy Requirements and Information Distribution

Privacy is a central issue in contact tracing platforms especially since the most essential data is the most sensitive data, e.g., geolocation of users, health records of carriers, business transactions. Societies have already witnessed the consequences of exposing sensitive information of participating individuals, e.g., [7].

Designers of contact tracing platforms need to consider the following three questions: (1) Where does the input data go? (2) Who performs the contact tracing computation on the input data? (3) Who is the recipient of the output of the contact tracing computation? A first attempt is to put all data into a single central server but such an approach introduces a single point of failure and *exposes sensitive data* to whoever has access to the server. Jumping ahead, our platform is based on cryptographic techniques and scalable secure protocols [8, 9, 10] that *do not reveal any information to the participating parties other than the result of the computation*.

Distributing the Result of Contact Tracing. The options for distributing the contact tracing results vary

depending on which party needs to be notified. We consider three major parties: the *medical authorities (MA)*, the *businesses*, and the *users*. There are two main goals for the BeeTrace platform:

- **Broadcast of High-Risk Places**. The MA publishes heat maps that depict the high-risk places. This list can help individuals to independently evaluate the risk of infection and self-report when in need for a medical test. Sharing this data with the community contributes in data-informed decisions about daily travel routes. Thus, users have the option to avoid high-risk areas and means of transportation.
- **Targeted Notifications**. The BeeTrace platform provides continuous updates for every participating user. It takes MA-reported diagnosis as input and distributes the tracing results to the relevant targets. Users that are part of the tracing result are notified by BeeTrace and can proceed with taking a medical test. We emphasize that the notifications are sent directly to the relevant party without exposing to MA the identity of the user at risk, which protects the privacy of BeeTrace users.

We propose a decentralized approach, which is a main key design of our BeeTrace platform: *a privacy-preserving* and *distributed query framework*, which is discussed in more detail in Section E. *Our privacy-preserving design focuses on introducing no extra privacy leakage, while improving the efficiency of the contact tracing.* We enable this functionality using secure multi-party computation (MPC) techniques which are shown to be practical in other deployed real-world systems.

Privacy Requirements. In addition to the information distribution requirements, one needs to consider privacy requirements such as, (1) *Anonymity requirement*: Avoid both explicit and implicit PII exposure. It should prevent linkage attacks and intersection attacks. (2) *Informed consent requirement*: Participants have the right to know what information is provided and how the information is processed. (3) *Confidentiality requirement*: All provided information should be protected against unauthorized access. (4) *Limited-time storage requirement*: All locally-stored information must not be kept longer than needed. (5) *Right to be forgotten requirement*: Participating individuals and businesses should be able to retract all their information generated through direct and indirect contact.

C BeeTrace: Platform Overview

The BeeTrace platform follows two new key components. The first component is a *unified data format with granularity adaptation* and introduces a new unified format that is comprised of a series of objects, see Figure 3. We address the challenges of multi-source contact tracing presented in Section B.1 by proposing this unified format which captures all the useful attributes that are generated by existing and new data sources. The second component is a *privacy-preserving distributed query framework* that allows the execution of queries over the objects of the unified data format. We note here that the result of each query is distributed only to relevant parties as described in Section B.2. To simplify the exposition in this section, we refer to unified data format and secure protocols without explaining the terms in detail. We note that a more thorough presentation of these two terms is provided in Section D and Section E.

In the following we present the workflow of the proposed platform BeeTrace. Our platform consists of two phases: the *setup phase* and the *operating phase*. The description of both phases is presented from the perspective of the three participating parties, the medical authorities (MA), the businesses, and the users.

C.1 Setup Phase

The first step that a party takes towards joining the BeeTrace platform is to download and install the *client* application of the platform according to the party's type. In particular, there are three types of client applications—MA-side client application, business-side client application, and user-side application. We assume that all parties

operate under the semi-honest security model and they use their real data as an input for the secure multiparty computation protocols.

Medical Authorities. In the setup phase, the medical authority (MA) simply installs the client of the platform. As a next step, the newly installed client converts all the information stored locally in the database of MA into the proposed unified data format. We note that the database of the MA at setup time contains all the verified infections that were collected *before* the MA joined the platform.

Businesses. The businesses that voluntarily participate in BeeTrace have to first install the business-side client application of the platform. For simplicity, we assume that there are no malicious actors that impersonate a business; we note that one can deal with such a scenario with known mechanisms such as Certificate Authorities. As a next step, the business converts locally each transaction that is stored in the local database into a unified data format entry. For example a hotel converts the stay of each visitor into a new unified data format entry. The rationale behind this step is that in future operating phases that take place *after the setup*, the hotel has already converted its information to the appropriate format and can participate in the privacy-preserving distributed query execution. We emphasize that the unified data format entries never leave the premises of the business and only participate in secure computation protocols that do not reveal the input of each party.

Users. The users that choose to participate in the platform have to first download user-side client application for their mobile device. As a next step the application converts the data collected from the device's sensors to the unified data format. For example, this data includes GPS traces and Bluetooth tokens. Similar to before, this conversion is performed so that the client can participate in the privacy-preserving distributed query execution and the plaintext unified data format entries never leave the premises of each user.

C.2 Operating Phase

In the operating phase, the participating parties collectively and continuously engage in privacy-preserving distributed query executions. The setup phase already converted the previously-generated data of each party to the unified data format. Every future data generation is converted to a unified data format entry automatically through the local client application. At a high-level, every discovery of an infection as an expansion of tracing starting points triggers the distributed execution of a query across all participants to finally generate the notifications. Our approach creates a *closed loop for continuous multi-source monitoring of the chain of infection*.

Medical Authorities. In the operating phase, the MA continuously receives reports about diagnosed carriers and contaminated places in our unified format. For each diagnosed carrier, the MA only learns a pseudonym or some tokens , i.e., an opaque ID or Rolling Proximity Identifiers. These unified data format entries in received reports are added to the list of tracing starting points. In turn the knowledge of the newly infected party triggers the execution of a query so as to discover the participating parties that came in direct and indirect contact with the carrier, i.e., discovery of tracing targets. Another responsibility of the MA is to collectively compute a high-risk heat map of the known infections so as to inform the community. The computation of this heat map takes place using cryptographic techniques that only output the aggregate information without revealing the identity of the infected individuals. The resulting map of this process is broadcast to all participating parties in the platform.

Businesses. The participating businesses use their locally stored data in unified format to contribute in the distributed privacy-preserving computation that is triggered by the contact tracing cycle. If a verified carrier completed a transaction with a participating business, then the platform notifies the manager of the place that the business is part of the list of tracing results. In turn the business shares the unified data format entries of the rest of the clients that performed a transaction within a certain time-frame. The affected users are now also part of the tracing results and, after further medical tests, the contact tracing continues. The business follows the direction of the MA to decontaminate its premises and removes itself from the tracing starting points list after the prescribed period of time. With the consent of the user, the business may also actively notify the user about their risk using communication methods already existing in their businesses.

Users. The participating users contribute in the distributed privacy-preserving computation with their data

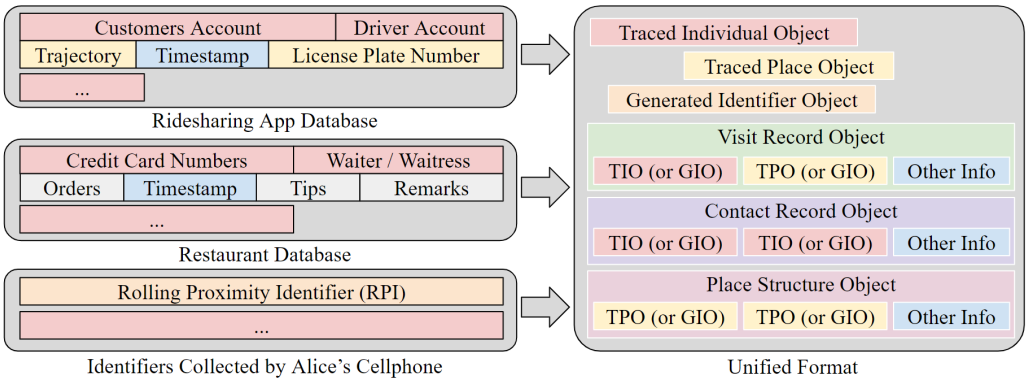


Figure 3: Format conversion between business data and our proposed unified format. On the left, we have records from businesses and third-party contact tracing apps with different format. All contact tracing related information from these records can be converted and stored locally in our unified format.

that capture, among other things, the GPS trajectories, collected Bluetooth tokens, visited businesses, precise QR-code generated geolocation etc. If a user is added to the tracing results via a confirmed direct or indirect contact then the platform directs the user to the closest medical test center. If the user is indeed a newly infected case, then the platform triggers another distributed query that updates the list of tracing starting points.

D Unified Data Format with Granularity Adaptation

In this section we give a detailed road-map of our vision about a unified data format. The first step towards breaking data silos is the creation of a *unified representation* for contact tracing data. We need to convert data from different sources to a universal format before processing it. Another challenge that we need to address is that spatial and temporal data is often in different granularity. Therefore, we need to create the corresponding abstraction to capture all generated data.

An Approach for Securely Unifying Data. For business purposes, the data stored locally at each business party typically contains information related to individuals, e.g., full name, username, registration email, credit card number etc. Our privacy requirements do not permit the uniquely identifiable information to leave the premise of the participating business/user. We introduce the notion of an *opaque ID* to represent each piece of sensitive information. This identifier does not reveal anything about the true identity of the entity it refers to. We note that opaque IDs can be generated by one-way hash function.

To demonstrate the *power of the proposed unified data format* we list a series of data objects that are relevant to typical contact tracing scenarios. We propose intuitive object descriptions to facilitate an easy conversion between original data and our proposed format. With the term *domain object* we denote a record associated with an opaque ID which in turn refers to either a user or a place. With the term *relation object* we denote a record that describes the relation between a pair of objects, e.g., the relation between a subway car and a subway train.

Domain Objects. We identify that the following domain objects are typically used in contact tracing:

- **Traced Individual Object.** It uses a set of key-value pairs of strings to describe the real-world information of an individual which will be used later in matching during contact tracing. It also contains a field for infection status and a field for risk evaluation status. The descriptive information can contain names, email addresses, phone numbers, or just usernames in apps depending on what information has already been collected for business purposes. All the values are secured by opaque ID to further protect user privacy. We note that a traced individual object may contain many fields, this design choice allows matching based on data that comes from different sources.

- **Traced Place Object.** These are the objects for traced places. They describe a static or dynamic place, and each object contains a field for contamination status. The description can be tailored to cover different location granularities, e.g., room of a hotel, floor of a hotel, hotel building, street of the hotel. We use place structure relation objects, detailed in the next paragraph, to explain the *belonging relation* between traced places.

- **Generated Identifier Object.** This object contains tokens generated by BLE or other contact tracing apps. These tokens can be associate to either individuals or places. Generated identifier objects are *ephemeral* and often have an expiration date different from the first two types of data objects. They also contain a field for risk evaluation. To illustrate its generation, when our client collects Rolling Proximity Identifiers from the Google/Apple solution we convert them to our unified format and use them for querying.

Relation Objects. These objects tie two distinct parties so as to keep track of either *direct contact* or *indirect contact*. We note that depending on the circumstances it is plausible that only one of the two parties generated a relation object. To overcome this asymmetry, we explain in the next section how to perform distributed queries to discover the relation objects that users/places unknowingly participate in. We identify that the following relation objects are typically used in contact tracing scenarios:

- **Visit Record Object**: This is a record that describes the event where an individual visited a place at a specific time (with varying precision granularity). It usually contains one traced individual object (or generated identifier object) and one traced place object (or generated identifier object). Visit record objects are created when businesses have an entry in their database describing this visit, e.g., a customer checking in at a hotel or a customer getting on a car. They are also created periodically by user-side client application using GPS sensors.

- **Contact Record Object**: This is a record that describes the event of two individuals being within a certain distance at a specific time (with varying precision granularity). It usually contains two traced individual objects (or generated identifier objects). For example, the ride-sharing companies create contact record objects for the driver and the customer, and the restaurants create them for customers who seat close. The user-side client application also collects generated identifier objects from existing contact tracing apps, e.g., ones that contain the Rolling Proximity Identifiers, and create contact record objects for them with the timestamp attached.

- **Place-Structure Object**: This record describes the relation between two traced places. For example, we want the system to understand a room in a hotel (which is a traced place) is in that hotel (which is another traced place). This comes handy, when we have a record of a carrier visiting the hotel without knowing which room (s)he stayed at. With place structure object, we can automatically infer that all traced places in this hotel are at high risk. Place structure objects form a tree-like structure for all traced places. E.g. room 301 is on the third floor of a hotel, which belongs to a certain hotel, which is part of a street block.

As discussed in Section C, each party will first download and install the client application of the platform. After the installation, the MA import all existing data about diagnosed carriers and contaminated places, and the MA-side client application will create the corresponding domain objects which are labeled infected. The business creates place-structure objects using the user interface of business-side client application and also import their contact tracing related data or connect its client application to its databases for continuous object creation. The user inputs their personal information and the user-side client application will generate traced individual objects secured by opaque ID for later matching. The user also turns on the sensors and grant app data access permissions so that user-side client application will start to collect and save domain objects and relation objects. It use GPS and QR code scanning to collect visit record objects, and use Bluetooth and other contact tracing apps

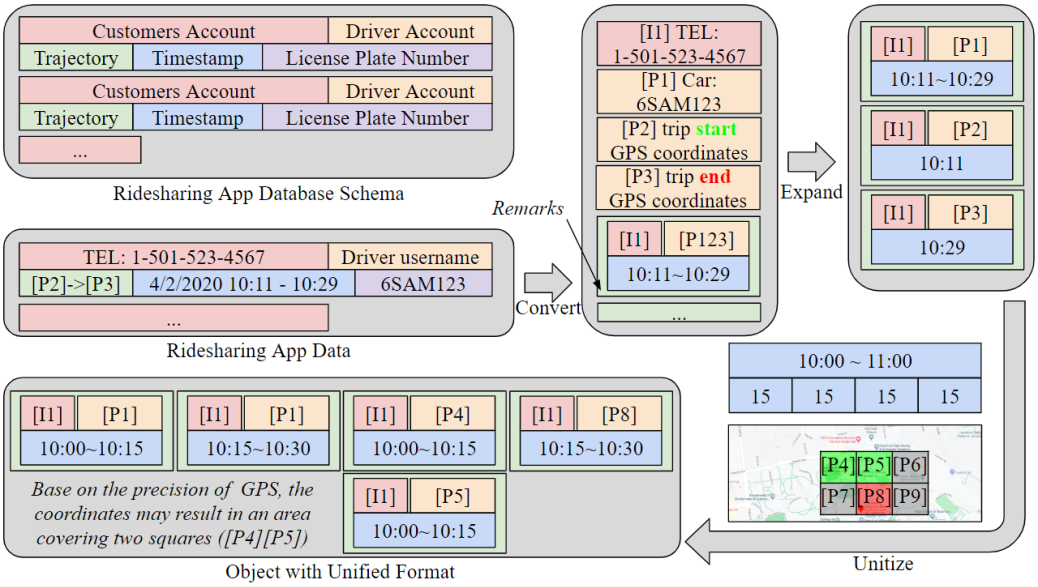


Figure 4: Illustration of expansion and unitization. The data from ride-sharing app is first converted into objects in our unified format. In the expansion step, the *visit record object* is expanded to three visit record objects that cover the start location, the end location, and the dynamic place of the car. Then in unitization step, we split one hour into four segments and generate the visit record objects accordingly.

to collect contact record objects, e.g., the Rolling Proximity Identifiers are first converted to generate identifier objects and are used for matching later.

We note that in each object description the *granularity* of the place and time may vary. We use two techniques to convert objects to the same granularity: *expansion* and *unitization*. Figure 4 presents a concrete example of these two steps using our motivating scenario with Alice.

In the *expansion step*, we expand an object by splitting the object into the smallest granularity. Given a traced place object, we analyze its tree structure implied by the place-structure objects to locate the smallest granularity. E.g., for GPS coordinates, we create a tree structure by superimposing grid cells (see the map in the lower right corner of Figure 4).

In the *unitization step* we process the continuous data entries, e.g., coordinates, timestamps. The continuous data may have a large domain; to limit the output space we split the domain into multiple segments and map the data accordingly. The number of segments is a tunable parameter of the platform. For example, in the figure we use fifteen minutes as the hour unit and split 24 hours into 96 segments with the same length. By processing the data under the same basic unit, we can seamlessly execute queries under a consistent domain of values.

E Privacy-preserving Distributed Query Framework

In this section, we introduce the proposed framework of platform BeeTrace for *privacy-preserving distributed data queries*. We first formalize the application scenarios for both targeted notification and high-risk place broadcast, then discuss security protocols that enable these applications.

In this work, we consider a *semi-honest* security setting where the adversary is assumed to follow the protocol, but attempts to infer additional information from the execution of the protocol. We assume there are multiple businesses and multiple users in our BeeTrace but only a single MA.

The Generated Identifier Objects, see Section D, may refer to either traced individuals or traced places. To illustrate this ambiguity, notice that a Bluetooth beacon installed at a place may use the same format as the Rolling

Proximity Identifier which is tied to individual and, consequently, we may not be able to distinguish these two scenarios. As a solution, when exchanging data related to either individuals or places, we assume Generated Identifier Objects to be both and always include them.

E.1 Targeted Notifications: A Privacy-Preserving Approach

There are two targeted notifications, a notification towards a business, and a notification towards a user. The first case takes place when the MA assists businesses in resolving whether they are a high-risk place. The second case takes place when the MA and businesses collectively use their data to assist a user in resolving whether they are in the tracing results. Concretely,

(1) **Notification Towards a Business.** The MA has domain objects that describe individuals and places in the tracing starting points. A business has a list of domain objects and wants to find out if any of its domain objects appear in the tracing starting points.

(2) **Notification Towards a User.** A user has a list of domain objects related to (or generated by) himself or the individuals (s)he met, e.g., the Rolling Proximity Identifiers from google/apple solution, and the places (s)he visited. The user wants to know if any of these objects matches the objects with infection risk or diagnosis from MA or businesses.

BeeTrace performs targeted notification while maintaining the privacy of each participant. To solve this problem in a privacy-preserving manner, we deploy an efficient cryptographic primitive called *private set intersection (PSI)*. The setup of PSI is the following: there are two parties, the sender and the receiver, and each party holds a set of elements. The PSI functionality allows the receiver to learn the intersection set and nothing else. Some real-world applications have already adopted the PSI primitive and achieve practical performance [9]. Using the PSI functionality, our BeeTrace can provide solutions for (1) and (2).

For (1), the MA and the business execute a PSI instance where the desired output of this protocol is the *tracing result*, i.e., the intersection between sets. Each party provides as an input a collection of locally-stored domain objects (e.g, traced place object). At the end of the PSI, the business who acts as PSI receiver learns the tracing result and nothing else, which protects the raw tracing starting points. On the other hand, the PSI gives nothing to the PSI sender MA which protects the business's input. Having the tracing result, the business can notify their customers about the newly discovered risk, e.g., the hotel in our example scenario, or just mark the status of their locally stored objects and wait for queries from users.

Similarly, for (2), the user executes a PSI instance with the MA as well as all businesses that (s)he has visited. With the list of domain objects as input from each party, PSI allows the user to learn if any of its objects appears in the object lists of MA or the businesses. Because of the cryptographic guarantees of the underlying PSI protocol, no information is revealed except the PSI output. If the intersection is non-empty then the user may potentially be infected. We emphasize that our platform also discover the risk that can be detected by different existing contact tracing apps. Because the generated identifier objects are also domain objects, and by matching them, e.g., matching the Rolling Proximity Identifiers, the user-side client application can deduce if the user is at risk.

As an alternative if we only want a business to check its contamination status *without revealing which customers was infected*, we can use a variation of PSI called PSI-CA [11]. From the output of this protocol the business only learns the number of infected users that visited its premises without identifying their opaque ID. An important technical detail of our approach is that we need to choose records with reasonable timestamp before issuing a query in order to reduce the false-positive alerts to customers, which is especially important when the query contains a large number of generated identifier objects describing ephemeral tokens like Rolling Proximity Identifiers.

When using PSI and PSI-CA to handle the generated identifier objects, the method is different from the google/apple approach but the ephemeral tokens, like Rolling Proximity Identifiers, are still used in a privacy-

- **The business knows the infected individuals**

  The MA confirmed that Bob is a diagnosed carrier therefore the MA keeps a record of him as a traced individual. After interacting with the MA, the ride-sharing company confirms that its customer Bob is infected (if we select to use PSI, not PSI-CA). Moreover the company learns that one specific car has been contaminated. This car is linked to Alice via her ride-sharing account history. Therefore, both the license plate number of the car and the name of Alice appear in the tracing result. When the platform app on Alice's cell phone runs query with the ride-sharing company, it discover that Alice is in the tracing results and displays a notification. In case the business initiates the discovery first, it will send Alice a notification using her account cell phone number and will contact the owner of that car and report that car as a contaminated dynamic place to MA.

- **The business does not know the infected individuals**

  MA confirmed that Bob is a diagnosed carrier and got a visit record from his cellphone. The restaurant chain discovered that one of its restaurants is contaminated based on the privacy-preserving queries with MA. If the restaurant does not keep records of all its customers, it cannot deduce who contaminated the restaurant and who is in risk of infection. Based on its staff rotation in the form of employee visit records the protocol outputs a new list of contaminated places. Since Alice has a visit record to one of these restaurants, during her periodical query she finds out that she is in the tracing result but nobody else knows about this.

Figure 5: Example for the privacy-preserving distributed query framework in BeeTrace

preserving way. The user-side application deduces the risk of individual infection by executing a PSI instance with the MA to check if the user has a matched token, which indicates whether the user has direct contact with someone in the tracing starting points. With tokens generated by Bluetooth beacons, this can be generalized to tracing for places similarly to cover indirect contacts.

Recall that the proposed platform BeeTrace aims to protect the privacy of participants while maintaining the contact tracing functionality. We illustrate this goal with two examples of distributed queries in Figure 5.

E.2 Private Discovery of High-Risk Place

To enable the discovery of high-risk places, the MA wants to learn which area the diagnosed carriers have visited while the carriers do not want to expose their own trajectories. In fact, the MA only needs the aggregated result, which does not expose information of an individual user. We first formally define the problem and then discuss a series of solutions that give different efficiency and privacy trade-offs.

- *Input:* We consider n users who tested positive and each of them has visited at most m places. We assume that each traced place object contains an opaque ID that refers to either a static place, e.g. a subway station, or to a dynamic place, e.g., a subway car. The identifiers are consistent across the platform so that all users have the same opaque ID for the same place.

- *Goal:* The MA wants to have daily access to the aggregate result of contaminated locations, which is the counting of related traced place objects. From previous findings in manual tracing results [12] it has been observed that it is typical to have several locations where a lot of carriers have passed by. The goal is to summarize this information concisely through a heat-map that describes the risk of each location.

In order to securely find high-risk places, we consider three *privacy-preserving solutions with different trade-offs*. The parameters that we consider are: the communication and time complexity of the protocol, the scalability

with respect to the number of users, and the security guarantees. Depending on the system specifications, the platform can be adjusted to the appropriate design. The three solutions are:

Voting Based Approach. If we consider users as voters and the MA as the party in charge of the tabulation, the setup resembles the electronic voting problem in which there are n voters, each have at most m vote ballots. The ballots are string identifiers representing candidates. The tabulation party wants to learn the aggregated result while the voters do not want to expose their own choice. Previous works build systems [13] and design protocols [14, 15, 16] to deal with this challenge. However, in the contact tracing scenario we have a significantly larger number of candidates to choose from, which can be a problem for both implementation and time/space complexity. To address the case where the number of candidates m is large, we explore an approach based on heavy hitters.

Heavy Hitters Based Approach. If we consider high-risk places as the frequent items from an open set, the setup resembles the heavy hitters problem. In particular, one is given a list of m integers in the domain $[1, \ldots, \ell]$, and the goal is to identify the items among the domain that appear frequently in the list. In BeeTrace, there are n users, each user has a set of visit record objects/items. If we consider the MA as the untrusted aggregator, the MA would like to continuously track the most recent heavy hitters (i.e. the popular items/places), namely those items that were held by at least a threshold number of different users. The problem of computing heavy hitters has been well-studied in the area of data stream algorithms. The earliest work can be traced back to the 1980s [17]. Over the last few years, several works propose efficient privacy-preserving schemes that can achieve practical performance [18]. However, most of the solutions rely on differentially private guarantees.

Oblivious Aggregation Approach. We are also interested in a secure computation model where no information about the private data held by the parties can be inferred during the execution of the protocol. Given an array of key-value pairs, oblivious aggregation allows to compute some aggregation function over all pairs with the same key in the privacy-preserving manner. We assume that there exists a one-to-one index mapping table of size ℓ between a particular place and the index (e.g. zip code area). Each diagnosed carrier has a set of (key, value) where the key is an index $i \in [1, \ldots, \ell]$ which has been mapped to a visit traced place in the mapping table, and the value indicates how many time the carrier visited this place. Given $n > 2$ diagnosed carriers (users), each holding a private set of key-value pairs, the functionality of oblivious aggregation is to allow the MA to compute the sum of all values with the same key without revealing any information. The oblivious aggregation [19, 10] can be implemented in time $O(\ell \log \ell)$ per each user's query.

F Conclusion

In this paper, we propose BeeTrace, a privacy-preserving contact tracing platform that breaks data silos. We show that by including business-side participation and standardizing the format of data, we can incorporate existing efforts and improve coverage. By deploying the state-of-the-art cryptographic protocols from the area of secure multi-party computation, we can achieve an efficient design that meets the privacy needs of contact tracing. We motivate our design choices by a complete workflow of a multi-source decentralized approach.

Our vision for a privacy-preserving data collaboration platform can be extended *beyond the problem for contact tracing*. The design principles of our platform can be adapted to address other pressing problems such as private contact discovery [9] as well as compromised credential checking [20]. The contact tracing scenario described in this work can be the first step towards a practical decentralized secure data collaboration platform.

Acknowledgments

We thank Peng Gao, Jiaheng Zhang, Tiancheng Xie, and Xinyun Chen for helpful discussion. This material is in part based upon work supported by the National Science Foundation(NSF) under Grant No. TWC-1518899, DARPA under Grant No. N66001-15-C-4066, Center for Long-Term Cybersecurity (CLTC), and IC3 industry

partners. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF, DARPA, CLTC or IC3.

References

[1] Katayoun Farrahi, Rémi Emonet, and Manuel Cebrian. Epidemic contact tracing via communication traces. *PLoS ONE*, 9(5):1–11, 2014.

[2] Otto Seiskari. BLE contact tracing sniffer PoC. `https://github.com/oseiskar/corona-sniffer`

[3] Christophe Fraser, Steven Riley, Roy M. Anderson, and Neil M. Ferguson. Factors that make an infectious disease outbreak controllable. *Proceedings of the National Academy of Sciences of the United States of America*, 101(16):6146–6151, 2004.

[4] Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science*, 6936(March):eabb6936, 2020.

[5] Apple & Google Privacy-Preserving Contact Tracing `https://www.apple.com/covid19/contacttracing`, 2020.

[6] Xinmei Shen From QR codes to social media, four ways China tracks Covid-19 `https://www.abacusnews.com/tech/qr-codes-social-media-four-ways-china-tracks-covid-19/article/3079595`, 2020.

[7] Nemo Kim in Seoul. 'more scary than coronavirus': South korea's health alerts expose private lives. `https://www.theguardian.com/world/2020/mar/06/more-scary-than-coronavirus-south-koreas-health-alerts-expose-private-lives`, 2020.

[8] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. PIR-PSI: Scaling Private Contact Discovery. *PoPETs*, 2018(4):159–178, 2018

[9] Daniel Kales, Christian Rechberger, Matthias Senker, Christian Weinert, and Thomas Schneider. Mobile private contact discovery at scale. *Proceedings of the 28th USENIX Security Symposium*, pages 1447–1464, 2019.

[10] T. H.Hubert Chan, Kai Min Chung, Bruce M. Maggs, and Elaine Shi. Foundations of differentially oblivious algorithms. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2448–2467, 2019.

[11] Ni Trieu, Kareem Shehata, Saxena Prateek, Reza Shokri, and Dawn Song. Epione: Lightweight Contact Tracing with Strong Privacy. Preprint arXiv:2004.13293 [cs.CR], 2020.

[12] NEWS WIRES. Ghana's president says one person infected 533 with Covid-19 at fish factory `https://www.france24.com/en/20200511-ghana-s-president-says-one-person-infected-533-with-covid-19-at-fish-factory`, 2020.

[13] Chris Karlof, Naveen Sastry, and David Wagner. Cryptographic voting protocols: A systems perspective. *14th USENIX Security Symposium*, pages 33–49, 2005.

[14] Jonathan Katz, Steven Myers, and Rafail Ostrovsky. Cryptographic counters and applications to electronic voting. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 2045, pages 78–92, 2001.

[15] C. Andrew Neff. Practical high certainty intent verification for encrypted votes. *VoteHere, Inc*, pages 1–24, 2004.

[16] David Chaum. True Voter-Verifiable Elections. *Iee Computer Society*, pages 38–47, 2004.

[17] Robert S. Boyer and J Strother Moore A fast majority vote algorithm. *Technical Report Technical Report ICSCA-CMP-32*. Institute for Computer Science, University of Texas, 1981

[18] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pages 2288–2296, 2017.

[19] Elette Boyle, Kai-Min Chung, and Rafael Pass. Oblivious parallel RAM and applications. In *Theory of Cryptography - 13th International Conference*, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II, pages 175–204, 2016.

[20] Lucy Li, Nick Sullivan, Bijeeta Pal, Rahul Chatterjee, Junade Ali, and Thomas Ristenpart. Protocols for checking compromised credentials. *Proceedings of the ACM Conference on Computer and Communications Security*, pages 1387–1403, 2019.

The Road for Recovery: Aligning COVID-19 efforts and building a more resilient future

Meredith M. Lee, Alicia D. Johnson, Katherine A. Yelick, and Jennifer T. Chayes
UC Berkeley Division of Computing, Data Science, and Society,
West Big Data Innovation Hub,
UC Berkeley Office of Emergency Management,
UC Berkeley Department of Electrical Engineering and Computer Sciences,
and Lawrence Berkeley National Laboratory

A A Pivotal Moment

Our society currently faces the most profound and deeply disruptive public health crisis in modern history. As communities across the world grapple with the COVID-19 pandemic, scientific advances spanning biochemistry and epidemiology to manufacturing and data engineering offer hope—and a spectrum of guidance is unfolding in an effort to respond to monumental shifts in our daily lives. The rising demand for data and the emerging efforts to responsibly collect, share, and analyze information across traditional boundaries play a vital role in our next steps.

From facilitating dialogue and decision making, to underscoring the importance of a shared, honest assessment of where we are in our collective fight against the pandemic, data are now more important than ever. Our computational capacity and the value that we as a community can generate through data science are fundamentally crucial to our resilience. As we look to transition from crisis response to longer-term recovery, we have an unprecedented opportunity to re-imagine a new data-enabled future.

B Emerging Research: The Challenge of the Unknown

Our landscape has shifted, and our paths forward will require unparalleled levels of creativity, persistence, and humility as we strengthen connections across sectors and disciplines, translating ideas into action. Although the data are rarely complete and extracting useful signals from the noise can be elusive, a number of COVID-19 data-enabled research efforts are emerging, including:

Analysis and Modeling

Considerable emphasis on data analysis and modeling has spurred discussion that bridges statistical methods, machine learning, and artificial intelligence with policy development and risk communication [1, 2, 3, 4]. For example, N. Alteri et al. have developed several predictors after curating available data, collaborating with

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

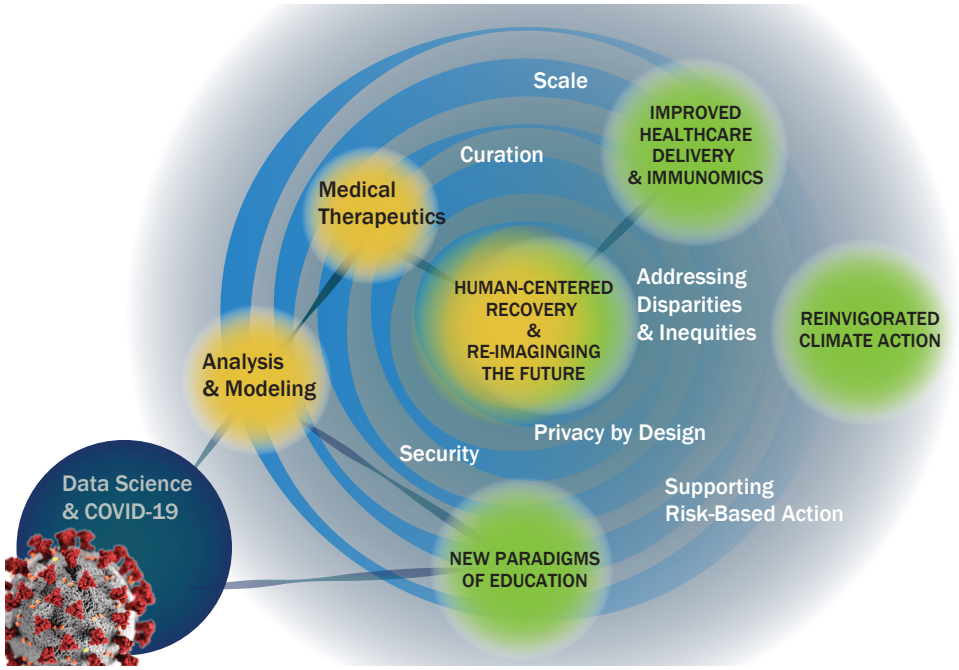


Figure 1: **Supporting Public Health.** The COVID-19 pandemic surfaces challenges where data science can enable new insights within a human-centered context. Current efforts, ranging from data analysis and predictive modeling to risk communication, set the stage for recovery and a new future, reinvigorated with critical questions and opportunities.

nonprofit organizations and healthcare providers to address medical supply needs for individual hospitals [5]. U. Seljak et al. have employed robust statistical methods to identify systematic errors and correct mortality rates that are integral to further analysis [6], while S. Yadlowsky et al. have modeled the infection prevalence of the virus [7], with J. Steinhardt and A. Ilyas noting shortcomings of existing tracking measures [8].

Such efforts can help communities visualize the nature of fluid events and iteratively explore reasonable response strategies when faced with unprecedented scenarios. Notably, as communities across the globe adjust their behavior against a backdrop of changing policies, guidance, and tactics, we have an opportunity to improve our assessments about the spread of the virus and to understand how our actions and other factors relate to key outcomes. Intentional and thoughtful data collection and analysis during recovery, focused on public health risks and implications—and acknowledging the critical lag between knowledge and informed action—will be of paramount importance.

Exploration to Aid Medical Therapeutics

As researchers and medical professionals from a variety of disciplines seek to accelerate actionable knowledge, collaborative frameworks and data exploration efforts are forming. National laboratories and science centers are searching databases for drug candidates that could be re-purposed for COVID-19 or used to inform clinical practice, sharing and updating progress more rapidly than traditional publication timeframes [9, 10]. Consortia aiming to speed dialogue and visualization of SARS-CoV-2 genomes through open online reports [11], as well as clinical study groups and collaborators working together in new ways, have the potential to shape and improve diagnostics, vaccine development, and outcomes [12, 13].

C Shared Human-Centered Context: The Need To Explore Difficult Questions, Together

As our collective response to the pandemic evolves, openly updating and sharing code, data, and insights as they develop is particularly critical for scientific reproducibility and effective coordination. Yet, a tension exists when evaluating the quality of information and choosing how to provide data, analysis, or tools to others. Noting the importance of clear, accurate, updated, and actionable information, the State of California coronavirus response team published a Digital Crisis Standard [14] with principles and framing questions that highlight accessibility, interoperability, and a focus on user needs and accountability.

Indeed, in the realm of disaster response and recovery, essential questions and unexplored frontiers arise in a fundamentally human-centered context. Issues stemming from ethics concerns and socioeconomic implications *must* be addressed early and often. Existing frameworks, theories, and practices that cut across disciplines are being put to the test and reconsidered as we face COVID-19, including:

Incorporating Privacy by Design

Adopted by the International Assembly of Privacy Commissioners and Data Protection Authorities nearly a decade ago, the foundational principles of the Privacy by Design framework [15] build upon views and practices emphasizing proactive, "by default", and embedded mechanisms that respect user privacy. With a backdrop of the General Data Protection Regulation and increasing prevalence of data privacy acts, laws, and norms [16, 17], emerging work towards privacy-sensitive proximity contact tracing looks to enable data exchange without compromising civil liberties, to query encrypted data, and to comply with unfolding guidelines for privacy safeguards [18].

Sharing information that could be relevant for individual or community health outcomes surfaces a number of deeper questions, where personal sentiment and expectations can vary widely. *What information should remain private, under what circumstances? What might I share in hopes of helping others? Will my privacy preferences change? Am I empowered to make choices about my privacy and data?* The tensions surrounding privacy and nuances around data sharing, while not new by any means [19, 20], are integral to our COVID-19 response and recovery.

Addressing Disparities and Inequities

With increased capacity to view data about disease incidence and mortality across demographic information, analysis showing disproportionate COVID-19 impacts on black communities, Latinx communities, and additional historically marginalized groups has surfaced [21]. Z. Obermeyer et al. have noted how existing measures can obscure rather than demonstrate inequality [22]. For example, communities with less access to testing for SARS-CoV-2 will have fewer diagnosed cases, making the epidemic look less severe. This can lead to disparities in attention and funding, and distort algorithms meant to help.

Networks of government cohorts have featured data-driven equity visualization tools and are increasingly convening public discussions focused on inclusion, a sense of belonging, and equity [23]. In order to support meaningful change, we will need to find ways to support empathy and shared understanding, directing energy towards measurable improvements. Partnerships to enable data compilation, analysis, and deeper research are essential, but we will need to leverage our ingenuity holistically and hold ourselves accountable to the bigger picture. *What levers do we have to address systemic issues? Who is missing from the conversation? How might we support public awareness, engagement, and education, creatively?*

Globally, with 265 million people projected to suffer from acute hunger by the end of this year, data-driven analysis is creating new options to help mitigate the most devastating impacts of COVID-19 in low- and middle-income countries. For example, applications of machine learning to satellite imagery and mobile phone data are

helping identify those individuals most in need of immediate humanitarian aid, and complement conventional methods that are limited by a lack of reliable and up-to-date data [24, 25].

Mechanisms for Supporting Risk-Based Action

Design through the lens of risk underscores the need to better quantify and qualify threats, vulnerabilities, and consequences. Throughout the pandemic, the corpus of available information and the assessment of risk are continuously changing as more is discovered about the virus and how it interacts with our communities. In a world where the opportunity for technology-driven situational awareness during a crisis is strikingly juxtaposed with an "infodemic" of extensive misinformation and dis-information [26, 27, 28], we are starving for transparent, appropriately vetted, and curated information in context.

In the clinical setting, aggregating data to capture patient risk indices could help communities more methodically assess and navigate difficult situations with new information, but effectively focusing attention and limited resources towards optimized patient outcomes remains an ongoing, and often overwhelming, challenge [29, 30]. In all too many cases, the pandemic is exacerbating existing vulnerabilities. Any "solutions" or approaches need to consider the nature of risks and how risk evolves over time.

D Compound Disasters and Interdependencies

Moreover, as the pandemic extends, the likelihood that additional communities will be impacted by both COVID-19 and compounded disasters—such as wildfires, hurricanes, earthquakes or tornadoes—is increasingly high. The U. S. Federal Emergency Management Agency (FEMA) has published guidance to prepare State, Local, Tribal, and Territorial (SLTT) responders to adapt their response and recovery procedures to this new combination of threats. The COVID-19 Pandemic Operational Guidance for the 2020 Hurricane Season (May 2020) notes the need for response and recovery adaptation that includes virtual damage assessments and significant reliance on the "whole of community" (including the private and non-profit sectors) to maximize their abilities to respond to simultaneously occurring disasters [31].

While in any given year, national-scale agencies managing emergencies routinely respond to multiple disasters simultaneously, SLTT responders often respond to one devastation at a time. Local economies strained by COVID-19 create an inability for communities to truly prepare and protect themselves from all impending hazards. Communities unable to protect themselves may suffer additional strain and damage that makes recovery from the compounded disaster even further from reach.

In an effort to proactively acknowledge limitations, the U.S. National Interagency Fire Coordination Center notes in its May 2020 plans, "In the event of a high disease spread scenario with a high rate of infection, the associated loss of individuals from service will severely tax the ability to maintain an adequate wildfire response, even during a moderately active fire season". The need for adaptation and re-orienting plans in light of COVID-19 extends to all hazards, as the methodologies and tools SLTT decision makers and responders currently use do not conform to physical distancing guidance and may amplify concerns about responder long-term health.

Moreover, some communities' plans and capabilities are not robust to damaged infrastructure or limited internet capacity, rendering strategies for multiple in-depth virtual interactions ineffective at best. Our reflections and how we navigate these challenges promise to impact our communities long after the virus has run its course.

While the movement to "flatten the curve" offered fairly uniform and straightforward guidance, to shelter in place and physically distance, the road for recovery and the notion of what a "new normal" looks like is riddled with additional interdependencies and tensions. These multifaceted challenges are increasingly being addressed through diverse coalitions, for example, with State-level economic development agencies working alongside university researchers, healthcare institutions, and volunteers to develop and share data, rapidly register products, and mitigate risks as guidance and authorizations evolve [32, 33].

E Re-imagining Resilience

With such a complex backdrop, it is more critical than ever to consider the roles for computing in social change [34] along with ethical implications [35, 36, 37, 38, 39] to enable response—responsibly. Data scientists and data enthusiasts globally have immense potential to light the path towards:

Improved Healthcare Delivery and Accelerated Growth of Immunomics

The pandemic has surfaced severe inadequacies in healthcare systems worldwide. New approaches to healthcare delivery must incorporate current lessons learned, and context, to address the fundamental needs of local communities and strive for measurable reductions in racial and economic biases. *How might we meet individuals and communities where they are, culturally and metaphorically? How can we imagine and achieve equitable health services across numerous barriers that exist today?* These framing questions should keep us grounded as exciting advances in computational immunomics promise to not only mitigate the effects of new viruses, but also provide novel treatments for autoimmune diseases, cancer, and other conditions.

Reinvigorated Climate Action

The dramatically altered global patterns of transportation and energy use during the pandemic will allow us to imagine new methods of interacting with each other and our environment. With many international and local borders closed for non-essential activity, and individual confinement extending, we have an opportunity to revisit the impacts of both policy and behavior changes on greenhouse gas emissions, our natural resources, and our clean energy economy [40]. Worldwide, individuals and organizations have a chance to reflect on our choices and climate commitments. Moreover, the pandemic continues to expose the lack of safe, accessible water in underserved rural as well as urban areas, raising opportunities to support equitable infrastructure investment and environmental justice.

New Paradigms of Online Education

Our concept of how we learn and the future of work may be forever changed. Online learning, once viewed as a less-prestigious option relevant only to a subset of students, is now at least temporarily the norm across the globe, for students from preschool through doctoral programs and in continuing and executive education. Ensuring that *all* students have access to appropriate hardware, software, internet connectivity, and resources to support distance learning is critical [41, 42, 43]. As remote teaching and learning extends, further research on the impacts of adopting new technologies—spanning issues from student privacy, safety, and mental health to the metacognitive aspects of learning, including motivation, resiliency, persistence, and scaffolding, will be needed. We must also be mindful of demands on instructors; training in new methods and logistical support for teachers should become a social priority. Experts, institutions, and authorities who are able to do so should share resources for effective and responsible distance learning widely, following open source models [44].

F Conclusion

While our community will surely grapple with unanticipated complexities over the next weeks, months, and years, we have a call-to-action in this moment and a hope for building a more resilient future. The future needs us to learn from the past and present—to deploy an inherently human-centered approach that connects research with the realities of crisis management, long-term recovery, and the vulnerabilities of being human. The future needs us to ask *Can we?* and *Should we?* as we learn how to harness the full potential of our collective resources.

Acknowledgements

The authors would like to acknowledge the dedicated individuals contributing to UC Berkeley's COVID-19 response, especially Jennifer Doudna, Michael Lu, Maya Petersen, Art Reingold, Anna Harte, Guy Nicolette, Fyodor Urnov, David Culler, and Joseph Gonzalez. In addition, M. Lee thanks collaborators from the Obama Administration White House Innovation for Disaster Response and Recovery Initiative and the National Science and Technology Council Subcommittee on Disaster Reduction. A. Johnson would like to thank the innumerable collaborations with local jurisdictions and communities that diligently remind her of the "why" behind the work of emergency management. Finally, the authors thank the National Science Foundation and network of regional data innovation hubs for efforts to spark and scale translational data science activities addressing societal needs.

References

[1] J. A. Lewnard et al., "Incidence, clinical outcomes, and transmission dynamics of severe coronavirus disease 2019 in California and Washington: prospective cohort study," *BMJ*, vol. 369, 2020.

[2] R. Best and J. Boice, "Where the latest COVID-19 models think we're headed - and why they disagree," May 2020. [Online]. Available: https://projects.fivethirtyeight.com/covid-forecasts/

[3] "Institute for Health Metrics and Evaluation." [Online]. Available: http://www.healthdata.org/

[4] "CovidActNow.org." [Online]. Available: https://www.covidactnow.org/

[5] N. Altieri et al., "Curating a COVID-19 data repository and forecasting county-level death counts in the United States," 2020. [Online]. Available: https://arxiv.org/abs/2005.07882

[6] C. Modi, V. Boehm, S. Ferraro, G. Stein, and U. Seljak, "How deadly is COVID-19? A rigorous analysis of excess mortality and age-dependent fatality rates in Italy," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/05/14/2020.04.15.20067074

[7] S. Yadlowsky, N. Shah, and J. Steinhardt, "Estimation of SARS-CoV-2 infection prevalence in Santa Clara County," *medRxiv*, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/03/27/2020.03.24.20043067

[8] J. Steinhardt and A. Ilyas, "Prevalence tracking mechanisms for SARS-CoV-2," (Accessed 29 May 2020). [Online]. Available: http://web.archive.org/web/20200529022542/https://www.stat.berkeley.edu/~jsteinhardt/publications/CovidPrevalenceTracking.pdf

[9] M. Smith and J. C. Smith, "Repurposing therapeutics for COVID-19: Supercomputer-based docking to the SARS-CoV-2 viral spike protein and viral spike protein-human ACE2 interface," 3 2020. [Online]. Available: https://chemrxiv.org/articles/Repurposing_Therapeutics_for_the_Wuhan_Coronavirus_nCov-2019_Supercomputer-Based_Docking_to_the_Viral_S_Protein_and_Human_ACE2_Interface/11871402

[10] S. H. Chen, M. T. Young, J. Gounley, C. Stanley, and D. Bhowmik, "Distinct structural flexibility within SARS-CoV-2 spike protein reveals potential therapeutic targets," *bioRxiv*, 2020. [Online]. Available: https://www.biorxiv.org/content/early/2020/04/18/2020.04.17.047548

[11] "Nextstrain.org," (Accessed 29 May 2020). [Online]. Available: http://web.archive.org/web/20200529023700/https://nextstrain.org/ncov-sit-reps/

[12] P. Schneider et al., "Rethinking drug design in the artificial intelligence era," *Nature Reviews Drug Discovery*, pp. 1–12, 2019.

[13] "kaggle CORD-19 Research Challenge," (Accessed 29 May 2020). [Online]. Available: http://web.archive.org/web/20200529015651/https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge

[14] "Digital Crisis Standard - covid19.ca.gov," (Accessed 15 May 2020). [Online]. Available: http://web.archive.org/web/20200515011426/https://github.com/cagov/covid19/wiki/Crisis-standard

[15] A. Cavoukian, "Privacy by design: leadership, methods, and results," in *European Data Protection: Coming of Age*. Springer, 2013, pp. 175–202.

[16] "General Data Protection Regulation." [Online]. Available: https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en

[17] C. D. Mares, "IoT: Privacy, security, and your civil rights," in *Women Securing the Future with TIPPSS for IoT*. Springer, 2019, pp. 15–36.

[18] G. Zanfir-Fortuna, "European Union's data-based policy against the pandemic, explained," (Accessed 29 May 2020). [Online]. Available: http://web.archive.org/web/20200529042315/https://fpf.org/2020/04/30/european-unions-data-based-policy-against-the-pandemic-explained/

[19] M. Langheinrich, "Privacy by design - principles of privacy-aware ubiquitous systems," in *Ubiquitous Computing*, G. D. Abowd, B. Brumitt, and S. Shafer, Eds. Berlin: Springer, 2001, pp. 273–291.

[20] L. Palen and P. Dourish, "Unpacking "privacy" for a networked world," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2003, p. 129–136.

[21] "The COVID Racial Data Tracker." [Online]. Available: https://covidtracking.com/race

[22] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan, "Dissecting racial bias in an algorithm used to manage the health of populations," *Science*, vol. 366, no. 6464, pp. 447–453, 2019.

[23] "Government Alliance on Race and Equity." [Online]. Available: https://www.racialequityalliance.org/

[24] J. Blumenstock, "Machine learning can help get COVID-19 aid to those who need it most," *Nature*, 2020.

[25] J. E. Blumenstock, "Fighting poverty with data," *Science*, vol. 353, no. 6301, pp. 753–754, 2016.

[26] H. Zade, K. Shah, V. Rangarajan, P. Kshirsagar, M. Imran, and K. Starbird, "From situational awareness to actionability: Towards improving the utility of social media data for crisis response," *Proc. ACM Hum.-Comput. Interact.*, vol. 2, no. CSCW, Nov. 2018. [Online]. Available: https://doi.org/10.1145/3274464

[27] "UN tackles 'infodemic' of misinformation and cybercrime in COVID-19 crisis." [Online]. Available: http://web.archive.org/web/20200529050148/https://www.un.org/

[28] S. J. Nightingale, M. Faddoul, and H. Farid, "Quantifying the reach and belief in COVID-19 misinformation," in press.

[29] F. Zhou et al., "Clinical course and risk factors for mortality of adult inpatients with COVID-19 in Wuhan, China: a retrospective cohort study," *The Lancet*, 2020.

[30] G. Onder, G. Rezza, and S. Brusaferro, "Case-fatality rate and characteristics of patients dying in relation to COVID-19 in Italy," *JAMA*, vol. 323, no. 18, pp. 1775–1776, 2020.

[31] U.S. Department of Homeland Security, Federal Emergency Management Agency, "COVID-19 Pandemic Operational Guidance for the 2020 Hurricane Season." [Online]. Available: https://www.fema.gov/media-library-data/1589997234798-adb5ce5cb98a7a89e3e1800becf0eb65/2020_Hurricane_Pandemic_Plan.pdf

[32] M. L. Zeidel, C. Kirk, and B. Linville-Engler, "Opening up new supply chains," *New England Journal of Medicine*, vol. 382, no. 21, p. e73, 2020. [Online]. Available: https://doi.org/10.1056/NEJMc2009432

[33] "California, Oregon, and Washington announce western states pact." [Online]. Available: https://www.gov.ca.gov/2020/04/13/california-oregon-washington-announce-western-states-pact/

[34] R. Abebe, S. Barocas, J. Kleinberg, K. Levy, M. Raghavan, and D. G. Robinson, "Roles for computing in social change," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020, pp. 252–260.

[35] J. Stoyanovich, B. Howe, S. Abiteboul, G. Miklau, A. Sahuguet, and G. Weikum, "Fides: Towards a platform for responsible data science," in *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, 2017.

[36] D. Gotterbarn, A. Bruckman, C. Flick, K. Miller, and M. J. Wolf, "ACM code of ethics: a guide for positive action," *Communications of the ACM*, vol. 61, no. 1, pp. 121–128, 2017.

[37] L. Erickson, N. Evans Harris, and M. M. Lee, "It's time for data ethics conversations at your dinner table," *Tech at Bloomberg*, 2018.

[38] Omidyar Network and the Institute for the Future, "The ethical operating system." [Online]. Available: https://ethicalos.org/

[39] N. Evans Harris, "Sharing data for social impact: Guidebook to establishing responsible governance practices." [Online]. Available: https://beeckcenter.georgetown.edu/wp-content/uploads/2020/01/Data-Sharing-Report.pdf

[40] C. Le Quéré et al., "Temporary reduction in daily global CO2 emissions during the COVID-19 forced confinement," *Nature Climate Change*, pp. 1–7, 2020.

[41] E. Lempinen, "The pandemic could open a door to new technology — and dramatic innovation — in education," 2020. [Online]. Available: http://web.archive.org/web/20200530235421/https://news.berkeley.edu/2020/05/27/the-pandemic-could-open-a-door-to-new-technology-and-dramatic-innovation-in-education/

[42] J. Burrell, "The value of the internet to rural populations: a case study from Mendocino County, CA," 2016. [Online]. Available: https://web.archive.org/web/20200530234610/http://www.mendocinobroadband.org/wp-content/uploads/Jenna-Rural-Internet-Report.pdf

[43] Z. Pardos, "What to look for in online learning apps for K-12," 2020. [Online]. Available: http://web.archive.org/web/20200530235325/https://gse.berkeley.edu/news/what-look-online-learning-apps-k-12

[44] United Nations Educational Scientific and Cultural Organization, "Supporting learning and knowledge sharing through open educational resources," 2020. [Online]. Available: http://web.archive.org/web/20200511005825/https://en.unesco.org/sites/default/files/covid19_joint_oer_call_en.pdf

flushleft]threeparttable
lined,boxed,vlined,ruled,linesnumbered]algorithm2e mycommfont export]adjustbox
OMSzplmmn
*arg max

DEEPEYE: A Data Science System for Monitoring and Exploring COVID-19 Data

Yuyu Luo† Nan Tang‡ Guoliang Li† Wenbo Li† Tianyu Zhao† Xiang Yu†
†Department of Computer Science, Tsinghua University ‡Qatar Computing Research Institute, HBKU
{luoyy18@mails., liguoliang@, li-wb17@mails., zhaoty17@mails., x-yu17@mails.}tsinghua.edu.cn, ntang@hbku.edu.qa

Abstract

The COVID-19 pandemic is a global health crisis of our time that has significantly affected almost every single person on earth in just several months. Even worse, we do not know when it will slow down and how long it will last. Analogous to fighting the COVID-19 pandemic in the physical world, data scientists need to deal with the infodemic of COVID-19 data to discover useful insight in order to guide wise and informative decisions, where the COVID-19 infodemic refers to all (messy) data relevant to COVID-19. In this paper, we present DEEPEYE*, an end-to-end data science system for monitoring and exploring COVID-19 data, which ranges from (task-driven) data preparation, (descriptive, diagnostic, and prescriptive) data analytics, user interactions through (linked) spatio-temporal data visualizations, and applications in different use cases.*

A Introduction

A.1 Where do we stand today on COVID-19?

The history of pandemics. Nothing has killed more people than infectious disease throughout the human history[1]. Many pandemics changed history; for example, the Antonine Plague (165-180) and the Black Death (1347-1351) have changed the history of Europe. Situations are getting worse in the last two decades, because epidemics happened much more frequently than what we have seen in history: SARS (2002-2003), Swine Flue (2009-2010), MERS (2012-present), Ebola (2014-2016), and now the COVID-19 (2019-present).

The history of COVID-19. Shortly after the first confirmed case in early January 2020, and a statement at January 21 from WHO's mission to China saying that there was evidence of human-to-human transmission, COVID-19 quickly spread out to almost every corner of the world. WHO officially named it a pandemic at March 11 2020. Till the date of June 1, 2020, there are more than 5.5 million confirmed cases and it has caused more than 350K deaths worldwide.

No system to stop a pandemic. Bill Gates envisioned, during his TedTalk on 2015[2], that if something will kill more than 10 million people in the next few decades, it will be infectious disease rather than wars. He has also questioned: "Are we ready for the next outbreak?" Sadly, the answer was not that the current health systems do not work for pandemics. Instead, there is *no global health system at all* for such pandemics.

Where do we stand today? Undoubtedly, COVID-19 has changed and will keep changing our history from many different dimensions, such as living style, studying style, the way of traveling, among many others. Indeed,

[1]https://www.visualcapitalist.com/history-of-pandemics-deadliest/
[2]https://www.ted.com/talks/bill_gates_the_next_outbreak_we_re_not_ready?language=en

we are not even at the peak of the 1st wave of COVID-19, and there might have the 2nd and the 3rd waves, such as the 1918 influenza pandemic and the 2009-2010 H1N1 pandemic, where the 2nd and 3rd waves were much more deadly than the 1st wave – we need to buckle up because the ride is just beginning. In order to stop or get better prepared to stop pandemics, global public health systems need to be ready, like military is ready for wars. Meanwhile, to help stop pandemics, *data science* has played a key role in discovering insights to guide decision makers and general people to make wise and informative decisions.

A.2 Do we have good data science systems for monitoring and exploring COVID-19?

From the real world pandemics to the virtual world infodemics. As the WHO Director-General Tedros Adhanom Ghebreyesus said[3]: "We're not just fighting an epidemic; we're fighting an *infodemic*." Here, the term "infodemic" generally refers to an excessive amount of information about a problem, which makes it difficult to identify a solution. Similar to the pandemics in the real world, infodemics are in the virtual world, where the goals of scientists in many domains are to deal with the information from the infodemics of the virtual world, so as to find meaningful insights that can be used to fight against the pandemics of the real world.

No data science system to stop an infodemic. Unfortunately, similar to the case there is no global health system that is ready to stop a pandemic, there is no data science system that is ready to deal with infodemics, even if we have seen the fruitful successes from many communities for data science, such as database, data mining, machine learning, natural language processing, bioinformatics, and many others. Essentially, all scientific methods are based on empirical or measurable *evidence* that is subject to the principles of logic and reasoning. In terms of infodemics, the evidence is the data that has been collected. However, the central problems are: (i) *Data is inaccurate:* the real confirmed cases and the number of death are conjectured to be much higher than the reported numbers. (ii) *Data is missing:* nobody knows precisely when and where did COVID-19 start, therefore many data is missing from its origin to the date that the data was first collected. (iii) *Data is inconsistent:* there are misinformation, disinformation, and rumors that are widely spread. (iv) *Data is not directly comparable:* different countries calculate mortality rate in different measures, *e.g.,* the death totals compiled by US CDC include 'probable' cases starting from April 16, 2020[4], while UK mainly considers the cases from the hospitals from the department and health of social care (DHSC) data[5].

The goal of an ideal data science system for infodemics. The *dark side* of infodemics is that *we never have the data well prepared for deriving the truth in any context*. Note, however, that the only certainty in (data) science is uncertainty, just like every single decision we have ever made. Hence, the *bright side of the dark side* is that this infodemics dilemma forces us to re-evaluate and re-design existing data science systems, for fighting against infodemics. Broadly speaking, the main goal of an ideal data science system for infodemics is about giving that data a purpose, which can provide hints to guide wise decisions. For example, although the reproduction number (Ro, pronounced R-nought or r-zero) of COVID-19 keeps changing, we are certain that *social distancing* is important. More concretely, an ideal data science system should be able to tackle all traditional data analytical tasks but under the situation of infomedics – data is very messy (the above i–iv) and new (and conflicting) data keeps coming – that can: (1) effectively collect, integrate and clean data from different sources; (2) make *descriptive analytics* to tell what happened in the past; (3) conduct *diagnostic analytics* to help understand why something happened in the past; (4) do *predictive analytics* to predict what will happen in the future; and (5) perform *prescriptive analytics* to recommend actions that one can take to affect those outcomes.

[3]https://www.who.int/dg/speeches/detail/munich-security-conference

[4]https://edition.cnn.com/2020/04/15/health/us-coronavirus-deaths-trends-wednesday/index.html

[5]https://www.health.org.uk/news-and-comment/blogs/understanding-the-data-about-covid-19-related-deaths

A.3 DEEPEYE: A small step towards a (ideal) data science system for infodemics

In this paper, we present DEEPEYE, an end-to-end data science system for collecting, cleaning, analyzing, and visualizing COVID-19 data. Since the system was launched in 05/02/2020, we have accumulated more than 2 million visits in a month. Some news media in China have reported our system[6][7], and we have launched online public courses to give tutorial for the system[8], which attracted nearly 100,000 people to participate. Besides, as a research-oriented platform, some research institutions (*e.g.,* CMU, The University of Hong Kong) also conduct preliminary processing and analysis of COVID-19 epidemiological data based on our platform.

Generally speaking, DEEPEYE consists of three layers: data preparation layer, data analytics layer, and user interaction layer (Section B).

In *data preparation layer*, a key observation we made is that it is impossible to prepare the data in the traditional way for some tasks, simply because data preparation is expensive and error-prone, especially when the data keeps changing every day. In particular, we will discuss task-driven data preparation, with the basic intuition that it is much cheaper to prepare the data that is needed for a given task (Section C).

The *data analytics layer* contains several components. For *descriptive analytics*, we use linked data visualization to provide sufficient context for a user to understand what happened in the past. We also use visualization recommendation techniques that can automatically discover interesting stories (Section D). For *diagnostic analytics*, we show that by combining with other (domain) knowledge, we can test our hypotheses (*e.g.,* the effect of urban (population) density or temperature to COVID-19) about why something happened (Section E). For *predictive analytics*, we have tried some predictive model, in particular Susceptible-Exposed-Infectious-Recovered (SEIR) [3], that has been widely used for epidemiology. However, our empirical results show that the prediction for COVID-19 is typically inaccurate that may due to the messy data of infodemics, hence we will not discuss further about it in this paper. For *prescriptive analytics*, we will discuss our collaboration with China mobile that help the government to recommend actions (Section F).

The discussion of *user interaction layer* will be blended with the discussion of data analytics layer, while describing various use cases.

B An Overview of DeepEye

B.1 System Architecture

We present DEEPEYE, an end-to-end framework to prepare data, select visualizations and allow easy-to-use interactions. An overview of DEEPEYE is given in Figure 1, which consists of three layers: *(task-driven) data preparation layer*, *(smart) data analytics layer*, and *user interaction layer*. Data preparation is responsible for crawling daily updated data from different sources, and cleaning them when needed (Section B.2). Data analytics describes the process of both which charts will always be shown (*e.g.,* a heat map on a world map showing new cases of every country), and how to automatically recommend visualizations that are "interesting" *w.r.t.* new incoming data, for visual analytics (Section B.3). Interaction allows a user to explore various and (maybe) new COVID-19 stories in an interactive fashion (Section B.4).

B.2 Task-driven Data Preparation Layer

This layer has a predefined pipeline to prepare data, which will run periodically, with the following three steps.

Data Collection. We collect data from the following data sources. (1) We download hourly the official data from the Chinese Center for Disease Control and Prevention (CDC) and other countries' CDCs. (2) We crawl infected

[6]https://news.tsinghua.edu.cn/info/1416/77464.htm

[7]https://www.aminer.cn/research_report/5e774aa1e34bad84366f1f7e?download=false

[8]https://www.bilibili.com/video/BV1FE411W7p4/?spm_id_from=333.788.videocard.0

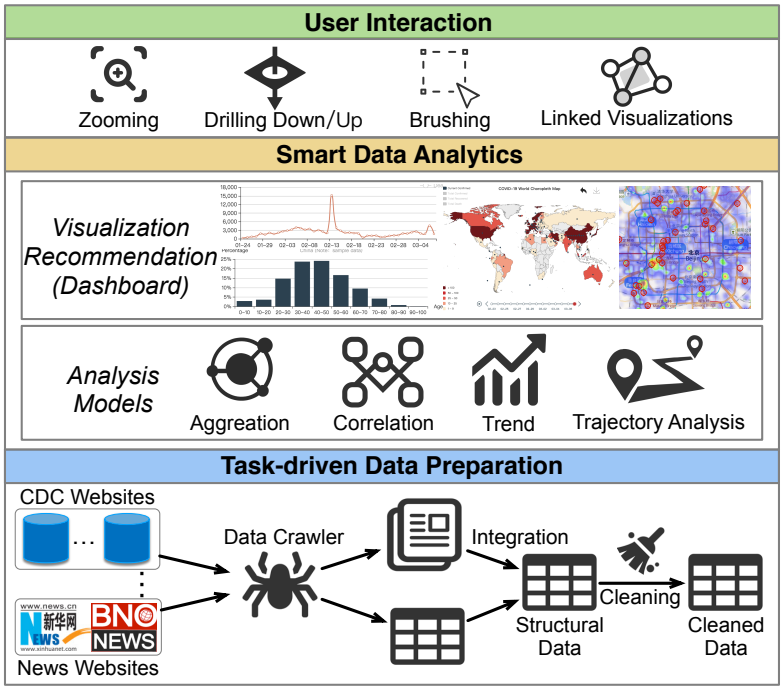


Figure 1: System Overview

cases' age and gender from authoritative news websites. (3) We also connect to other data sources like population statistics, temperature data, and so on. (4) We are also provided with trajectory data of (potentially) infected persons from China Mobile Limited[9].

Data Integration. Next, we need to integrate different types of data into predefined relational tables (*i.e.,* global views). For example, we need to extract report date, location, patients' type, #-cases from each country's CDC's reports, and perform schema alignment into $S1$: (*Date, Country, State/Province, City, Total Confirmed, Active Confirmed, Total Deaths, Total Recovered, Death Rate, Recovered Rate, Gender)*, a typical ETL-based data integration process.

Data Cleaning. After integrating data from multiple sources, there have typical data errors such as duplicates, missing values, synonyms, and so on. Because data cleaning is known to be tedious and error-prone, we employ our recently proposed technique VISCLEAN [7] for visualization-aware data cleaning, which is way cheaper than cleaning the entire dataset. This is doable only after the charts to display have been selected, as discussed below. We will depict more details about visualization-aware data cleaning in Section C.

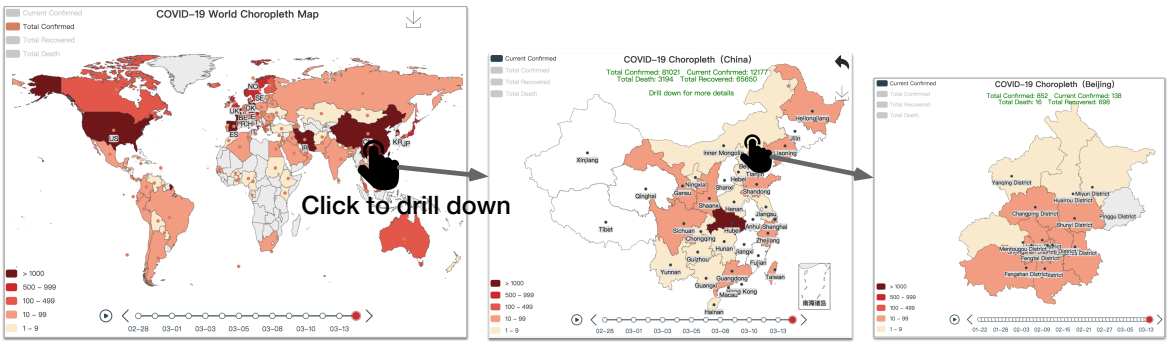
B.3 Smart Data Analytics Layer

Based on the availability and reliability of data and meta-data, we have successfully conducted the following types of data analytics.

Descriptive analytics. We use linked data visualization and visualization recommendation algorithms to effectively show what happened in the past.

Diagnostic analytics. We use maps with different layers to test the spatio-temporal properties of COVID-19 data, especially to show the effect of urban (population) density and temperature to the outbreak of COVID-19.

[9]We are collaborating with the company and got mobile phone location data under privacy protection.



(a) world-level (b) country-level: China (c) province-level: Beijing

Figure 2: Drill Down Operation

Prescriptive analytics. Based on the collaboration with companies to get private data, we were able to do some meaningful prescriptive analytics that can recommend actions to decision makers.

B.4 User Interaction Layer

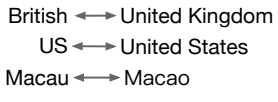
This is the interface we present to the public. When a user visits DEEPEYE, he/she can further explore visualizations by interactive module for finding more interesting insights. DEEPEYE supports popular interactions such as drilling down/up, zooming in/out and linked visualizations, powered by the visualization library ECharts [5].

Take drill down as an example (see Figure 2), when a user clicks a country (*e.g.,* China) on the world-level map, the map will drill down into the country-level map for more details. Note that, DEEPEYE provides linked visualizations of the analytical results. That is, when a user performs a drill down operation, other visualizations will also drill down into certain level automatically. In addition, the user can zoom in/out the map by rolling up/down the mouse.

C Task-driven Data Preparation

In the era of big data, from the data perspective, the data can come from governments, business companies, Web, and so on. As shown in Figure 1, in the scenario of COVID-19, the data can come from WHO, each country's Center for Disease Control and Prevention (CDC), news websites, microblog text, and other statistic datasets (*e.g.,* population, temperature and international airline data). In addition to heterogeneous data sources, how to integrate data with different data formats (*e.g.,* table, JSON-like, and text) from different data sources is another problem. One concrete example is how to extract the data from the CDC daily reports and then align such data into predefined relational tables. In this work, we develop an algorithm to crawl, extract data and fill them into relational tables.

Clean data is one of the essential requirements for data analytics and accurate analysis results [1, 2]. However, data collection and integration usually introduce errors (*e.g.,* missing values, mixed formats, and duplicates) in data due to the integration of heterogeneous data sources [6]. As shown in the Figure 3(a), they are four types of common data errors in the scenario of COVID-19. First, the synonyms usually appear when integrating data from different sources, especially when the data standard of data sources may change. The second data errors is missing value, as shown in Figure 3(b), this type of data error is usually caused by the data source not being updated in time. More concretely, on the 05/02, the data source may not have published the total number of deaths on that day, so a missing value was introduced. The missing value imputation is usually estimated using the data of the first few days and later, or imputation by querying other data sources. Similar to the missing value, as shown in Figure 3(c), it may also introduce incorrect values. We use the average values of the data of the first few days and later as a repair candidate to correct the wrong value. As shown in Figure 3(d), we crawl the



(a) synonyms

Date	Total Deaths
05/01	230
05/02	Missing
05/03	231

(b) missing values

Date	Total Cases
02/21	591
02/22	659
02/23	620

(c) incorrect values

地区 (City)	性别 (Gender)	年龄 (Age)	行程描述 (Trajectory Description)
蚌埠 Bengbu	男 Male	45	经营一家房地产中介公司，常年居住在武汉市江岸区 Operates a real estate agency company and lives in Jiang'an District of Wuhan City
蚌埠 Bengbu	女 Female	49	现住址为蚌埠市经开区长淮卫镇卫东村 The current address is Weidong Village, Changhuaiwei Town, Jingkai District, Bengbu
蚌埠 Bengbu	男 Male	45	患者为上海铁路局集团公司乘务员，住蚌埠市经开区湖滨社区C区4栋。工作线路：蚌埠–北京西(K148, K147) Working for Shanghai Railway and living in Building 4, District C, Hubin Community, Jingkai District, Bengbu City. Journey: Bengbu Station–Beijing West Station(K148, K147)
蚌埠 Bengbu	女 Female	42	疑似居住于田家庵区 Suspected of living in Tianjiaan District
蚌埠 Bengbu	男 Male	45	家住蚌埠市经开区湖滨社区C区4栋，上海铁路局集团公司员工。工作线路：蚌埠–北京西（K148, K147) He lives in Building 4, District C, Hubin Community, Jingkai District, Bengbu City. He works for Shanghai Railway, Journey: Bengbu Station–Beijing West Station(K148, K147)

(d) duplicates

Figure 3: Examples of Data Errors

travel records of some infected people from the news website. However, because data is crawled from multiple websites and then integrated, it is easy to introduce duplicate records. For duplicate records, we can use the entity matching algorithm to detect and remove duplicate records.

Because the analysis scenarios of COVID-19 have high requirements on data quality, we first need to ensure the data quality, and then consider how to reduce the cost of data cleaning. However, it's too expensive and infeasible to resolve all errors. Therefore, we discuss how to conduct problem-oriented (task-driven) data cleaning for COVID-19. The key idea is that we only clean those data relevant to the data analytics tasks, which can reduce the cost of data cleaning. In other words, instead of applying cleaning tools to clean each type of data errors before visualization, we aim to clean those data errors that have heavy impact on the accuracy of the visualization results. We devise our recent technique, a framework for visualization-aware data cleaning, called VISCLEAN to achieve this goal. The key idea is that we first generate visualizations based on the possible dirty dataset, and then run cleaning tools in the backend to find possible errors and their repairing candidates. Next, we organize those data errors and their repairing candidates by a graph model. Based on the graph and a predefined benefit model, we aim to select the most "beneficial" questions to interact with the user. The large the benefit is, the high quality of visualization improvement is. After the user answer the data cleaning questions, the system will fix data errors and refresh the visualization.

D Descriptive Data Analytics of COVID-19

Visualization selection generates three categories of charts: *linked* common visualizations, *ad-hoc* visualizations, and *recommended* visualizations.

(General) Linked Visualizations. There are common visualizations for spatio-temporal data exploration, such as a choropleth map (a heat map based on a map), line charts to show various trends, bar charts to show the comparison between various groups, scatter charts (or bubble charts) to quantify the relationship between two quantitative variables (*e.g.,* death rate vs. cure rate). We carefully selected charts (see Figure 4) that can attract a wide range of interest, and make them "linked", *e.g.,* when one zoom in from a world level to a country level,

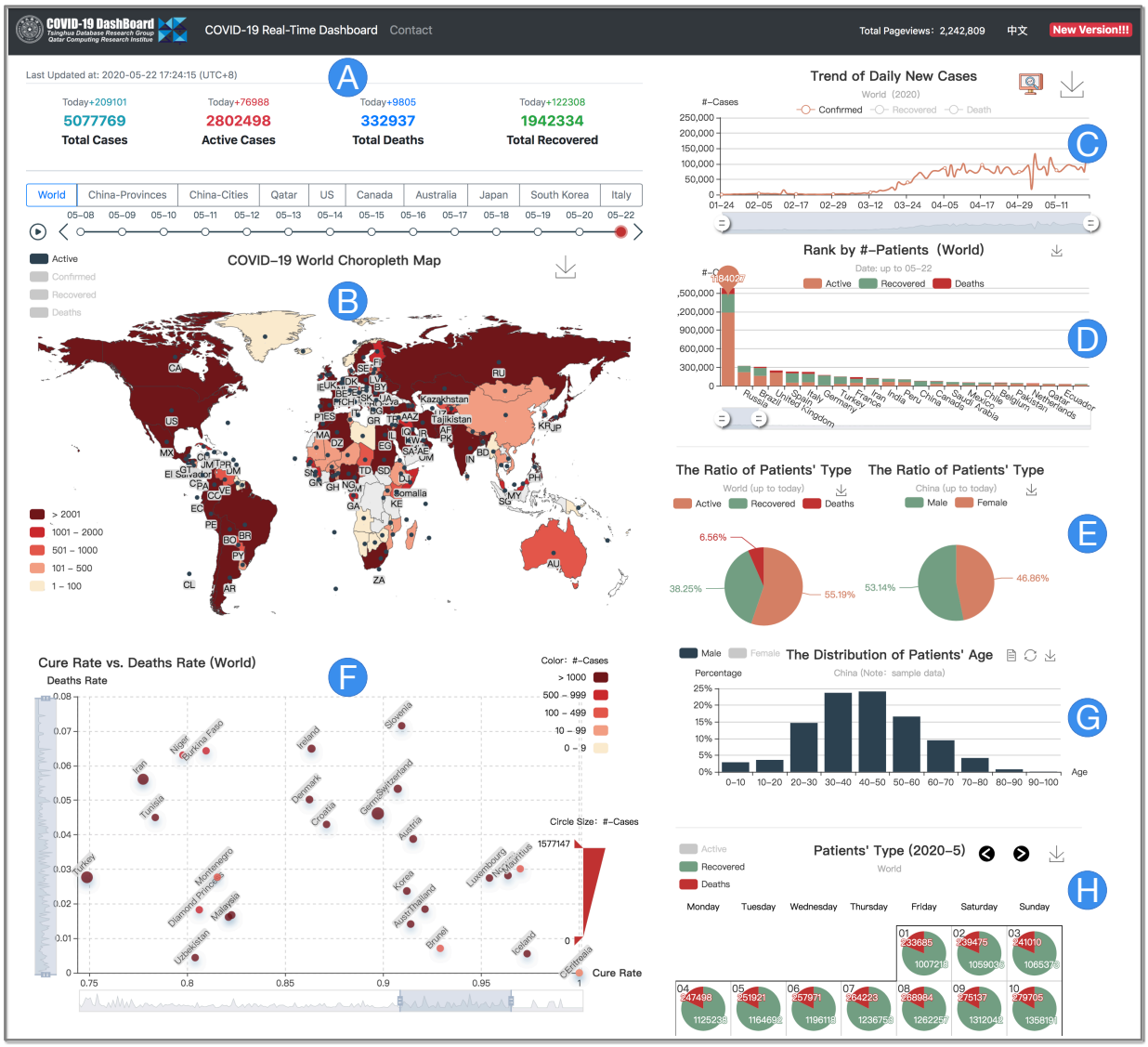
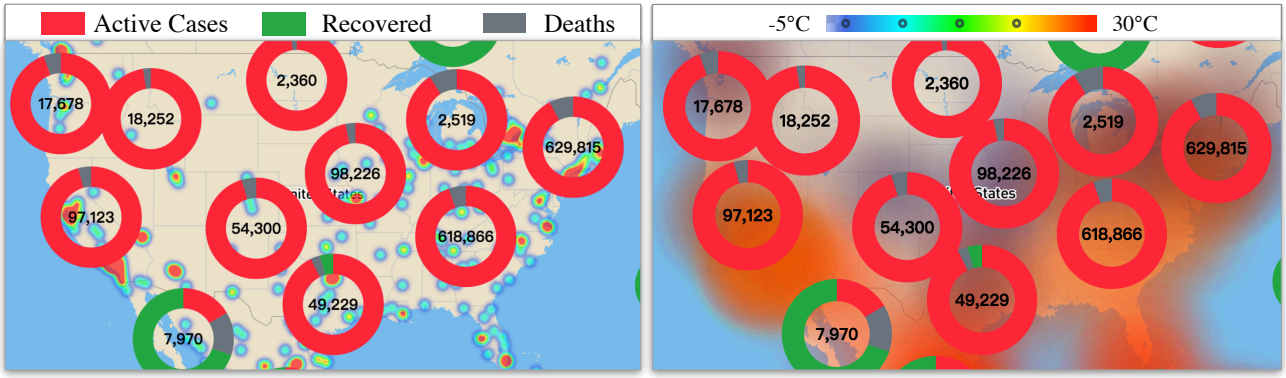


Figure 4: The Frontend of DEEPEYE (https://ncov.deepeye.tech/en)

all the other charts will be zoomed in, so as to provide a synchronized view from multiple charts. The user can get high-level situations of COVID-19 from Figure 4. For example, the user can catch the overall information of the reported cases from Figure 4(A). The choropleth map in Figure 4(B) shows the location and number of confirmed cases, deaths and recoveries for all affected countries. It also provides a timeline toolbar for the user to look back upon previous situations, and a user can click the “” button to show an animation. The user can click a country, *e.g.,* China, to drill down into the country-level (province-level or city-level) for more details. Since we apply the linked visualization techniques, the rest of the visualizations will also drill down into the country-level. Figure 4(C), a line chart, illustrates the daily increased cases of the selected location. The stacked bar chart in Figure 4(D) depicts the number of cases for the selected location. The pie charts in Figure 4(E) show the proportion of patients' type. Figure 4(F), a bubble chart, illustrates the relationships across #-cases, deaths rate, and cure rate. The bar chart in Figure 4(G) shows the distribution of patients' age, and the calendar chart in Figure 4(H) illustrates the proportion of types of reported cases for each day.

Location-based Search. For the general public, DEEPEYE provides the module of finding confirmed cases in



(a) the effect of urban (population) density (b) the effect of temperature

Figure 7: Diagnostic Data Analytics (Case in United States)

nearby neighborhoods. Take Figure **??** for example, users can understand the COVID-19 situations near *Tsinghua University, Beijing, China* by a location search box. Note that this module only supports for the Mainland China area currently.

Similarity Trends Discovery. DEEPEYE also supports the similar trend search functionality for finding similar trends. For example, if the user wants to find those trends of confirmed cases that are similar to *Switzerland*, the similarity search functionality will return top-k similar trends about *Switzerland*. The running example is shown in Figure **??**. Besides line charts, the similar trend search also supports other charts (*e.g.,* bar chart and pie chart). Thanks to this functionality, users can perform comparative analysis easier.

Automatic News Generation. Automatic news generation, in other words, automatically extracting insights from data visualization is promising research and practical direction [10]. Currently, the user usually interacts with the visualization dashboard to get insights and make decisions. For example, the reporter may interact with the dashboard to observe the trend of daily new confirmed cases of each country/state and find a set of similar trends (or find a set of rapidly increasing trends) as news stories. In this scenario, it heavily relies on the user to manually get insights and write a news release. One intuitive idea is whether we can derive insights (news stories) from the visualization dashboard automatically. Roughly speaking, given a set of visualizations $\mathbf{V}$ and a news generation model $\mathbf{M}$, the automatic news generation problem is to output a set of new stories $\mathbf{S}$. The key challenge is how to design the news generation model $\mathbf{M}$. One straightforward approach is predefined a set of expert knowledge rules to mine insights from the visualization dashboard. Such rules can be similar trends discovery, outlier trend detection, trends comparison, and so on.

E Diagnostic Data Analytics of COVID-19

Another purpose of data visualization is to perform hypothesis testing, we show how to design visualizations to test two hypotheses – urban (population) density *vs.* total confirmed cases, and temperature *vs.* total confirmed cases.

Urban (Population) Density. One intuitive hypothesis is that whether high population densities catalyze the spread of COVID-19? Since we want to know the relationship (correlation) between the population density and the spread of COVID-19, we first visualize the population density in the map named *population density map*, and then we map the confirmed cases on the top of *population density map*. As shown in Figure 7(a), it takes United States as an example. It shows that in areas with high population density and without lockdown policy, *e.g.,* New York and California, more people are infected with coronavirus. For example, New York with a relatively high population density is likely more vulnerable to the spread of the coronavirus. This conclusion is reasonable, because the intensive contact greatly increases the probability of coronavirus transmission [11].

Figure 8: Tracking Infection Path

Temperature. We also design visualization to show the relationship between the outbreak of COVID-19 and the temperature factor. Similarly, we first visualize heatmap using temperature data, and then we map the confirmed cases on the top of the heatmap. As shown in Figure 7(b), it is hard for us to make conclusions like the higher temperature, the more infected cases, or the lower temperature, the less infected cases. For example, the average temperature in the central United States is lower than in California, but there are also hundreds of thousands of infected people in the central United States. Comparing Figure 7(a) and Figure 7(b), we can find that under the background of no lockdown, the population density has a stronger correlation with the number of confirmed cases.

F Prescriptive Data Analytics of COVID-19

We also design ad-hoc visualizations to answer specific questions. In terms of COVID-19, besides publicly available datasets, we also have private trajectory data of potentially infected persons. Based on which we have designed two map-based visualizations, one to show infection paths of these patients (see Figure 8), and the other to show the level of risk for each area (see Figure 9) and thus suggest the authorities to take different anti-epidemic policies for different areas.

F.1 Infection Path

Based on the trajectory data of (potentially) infected persons, we can support to visualize and track the infection path at the high-level. Taking Figure 8 as an example, a person started from *Beijing Haidian Hospital* at 13:28 on 2020-02-28, and walked through several streets and finally arrived at his/her neighborhood. Therefore, we cluster those trajectories of infected persons to find a group of high-risk roads. Moreover, we devise a trajectory similarity search technique to find other trajectories similar to those trajectories of infected persons. It will help us to find and report the potentially high-risk groups. Thus, the authorities can take different anti-epidemic policies against people at different risk levels.

Findings and Insights. According to the sample trajectory data of (potentially) infected persons provided by China Mobile Limited, we find that most of the trajectories passed through some living places such as restaurants and supermarkets, and finally return home. Some trajectories intersect with public transportation such as trains and subways. There are a few trajectories with several other trajectories that coincide, indicating that they may be traveling together. Since many trajectories have visited supermarkets and other living places, the government can

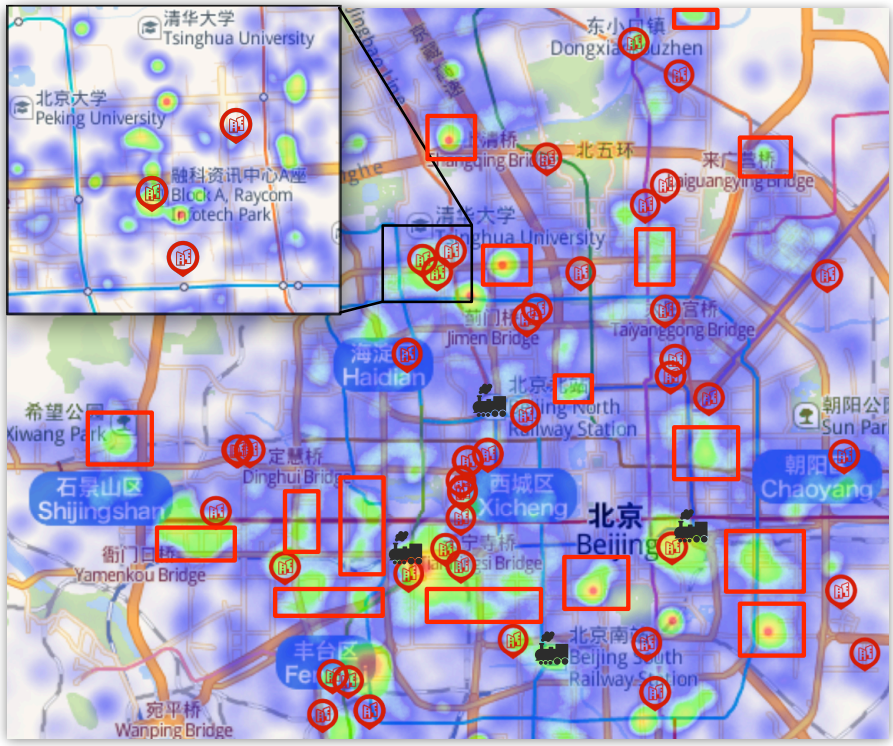


Figure 9: High-risk Area Discovery

suggest that people go to supermarkets as little as possible, or that different people go to supermarkets at different times, and limits the number of customers in supermarkets.

F.2 High-risk Areas Discovery

To further explore the trajectory data of (potentially) infected persons, we also design a visualization for the trajectory points using a heatmap. Figure 9 gives an at-a-glance understanding of the spatial distribution of the potentially infected persons from Hubei, China to Beijing, China. Those "hot" areas mean there have more potentially infected persons visit. We also indicate those neighborhoods that have several confirmed cases in the heatmap by the symbol .

Findings and Insights. According to the location data of potentially infected persons from Hubei to Beijing provided by China Mobile Limited, we make the following observations: (1) Most of the potential persons arrive in Beijing through transportation hubs such as railway stations (in Figure 9) and airports. Intuitively, such transportation hubs likely play a significant role in the spread of the coronavirus in Beijing City. Therefore, the government should take corresponding measures for these transportation hubs to minimize the risk of the transmission of coronavirus; (2) Many potentially infected persons visit some commercial places (*e.g.,* supermarkets) and end up staying in their residence; and (3) There is a high overlap between areas where high-risk people are active and neighborhoods containing confirmed cases. Intuitively, we can infer that other places (*e.g.,* the red rectangles) visited by high-risk groups may also be dangerous. Therefore, the authorities can take different anti-epidemic policies against people at different risk levels. Note that, for those "hot" areas do not appear confirmed cases (*e.g.,* the red rectangles), some of them are later reported confirmed cases by the government. Thus, the authorities can take precautions against these areas in advance and take different anti-epidemic policies against different areas to achieve refined and effective management.

G Concluding Remarks

There exist many systems for monitoring and analyzing spatio-temporal data, such as a dashboard for visually tracking the outbreak of COVID-19 [4] and a tweet stream sentiment analysis system for US election 2016 [9]. One lesson from the existing systems is that they are usually designed on a case-by-case basis and built from scratch, which cannot fully leverage the recent techniques for data integration and automatic visualization.

On the one side, DEEPEYE-COVID-19 shares many common visualizations as the other popular websites for tracking COVID-19 cases. On the other side, it differs from the others in (1) DEEPEYE-COVID-19 is based on a general end-to-end framework DEEPEYE, and leverages recent techniques for data preparation (*e.g.,* VISCLEAN [7]) and for visualization recommendation (*e.g.,* DEEPEYE [8]); (2) it supports linked visualization for the users to easily zoom in/out multiple visualizations by a single click; and (3) it also obtains some private data that is not publicly available, so it can demonstrate some unique features.

References

[1] Z. Abedjan, X. Chu, and et.al. Detecting data errors: Where are we and what needs to be done? *PVLDB*, 9(12):993–1004, 2016.

[2] C. Binnig, L. D. Stefani, T. Kraska, E. Upfal, E. Zgraggen, and Z. Zhao. Toward sustainable insights, or why polygamy is bad for you. In *CIDR*.

[3] J. M. Carcione, J. E. Santos, C. Bagaini, and J. Ba. A simulation of a covid-19 epidemic based on a deterministic seir model. *arXiv preprint arXiv:2004.03575*, 2020.

[4] E. Dong, H. Du, and L. Gardner. An interactive web-based dashboard to track covid-19 in real time. *The Lancet Infectious Diseases*, 2020.

[5] D. Li, H. Mei, Y. Shen, S. Su, W. Zhang, J. Wang, M. Zu, and W. Chen. Echarts: A declarative framework for rapid construction of web-based visualization. *Vis. Informatics*, 2(2):136–146, 2018.

[6] G. Li. Human-in-the-loop data integration. *Proc. VLDB Endow.*, 10(12):2006–2017, Aug. 2017.

[7] Y. Luo, C. Chai, X. Qin, N. Tang, and G. Li. Interactive cleaning for progressive visualization through composite questions. In *ICDE*, 2020.

[8] Y. Luo, X. Qin, N. Tang, and G. Li. DeepEye: Towards Automatic Data Visualization. In *ICDE*, 2018.

[9] D. Paul, F. Li, M. K. Teja, X. Yu, and R. Frost. Compass: Spatio temporal sentiment analysis of US election what twitter says! In *SIGKDD*, 2017.

[10] X. Qin, Y. Luo, N. Tang, and G. Li. Making data visualization more efficient and effective: a survey. *VLDB J.*, 29(1):93–117, 2020.

[11] J. Rocklöv and H. Sjödin. High population densities catalyse the spread of covid-19. *Journal of travel medicine*, 27(3):taaa038, 2020.

Data Engineering

It's FREE to join!

TCDE

tab.computer.org/tcde/

The Technical Committee on Data Engineering (TCDE) of the IEEE Computer Society is concerned with the role of data in the design, development, management and utilization of information systems.

- Data Management Systems and Modern Hardware/Software Platforms
- Data Models, Data Integration, Semantics and Data Quality
- Spatial, Temporal, Graph, Scientific, Statistical and Multimedia Databases
- Data Mining, Data Warehousing, and OLAP
- Big Data, Streams and Clouds
- Information Management, Distribution, Mobility, and the WWW
- Data Security, Privacy and Trust
- Performance, Experiments, and Analysis of Data Systems

The TCDE sponsors the International Conference on Data Engineering (ICDE). It publishes a quarterly newsletter, the Data Engineering Bulletin. If you are a member of the IEEE Computer Society, you may join the TCDE and receive copies of the Data Engineering Bulletin without cost. There are approximately 1000 members of the TCDE.

Join TCDE via Online or Fax

ONLINE: Follow the instructions on this page:

www.computer.org/portal/web/tandc/joinatc

FAX: Complete your details and fax this form to **+61-7-3365 3248**

Name _____

IEEE Member # _____

Mailing Address _____

_____

Country _____

Email _____

Phone _____

TCDE Mailing List

TCDE will occasionally email announcements, and other opportunities available for members. This mailing list will be used only for this purpose.

Membership Questions?

Xiaoyong Du
Key Laboratory of Data Engineering and Knowledge Engineering
Renmin University of China
Beijing 100872, China
duyong@ruc.edu.cn

TCDE Chair

Xiaofang Zhou
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
zxf@uq.edu.au

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314

Non-profit Org.
U.S. Postage
PAID
Los Alamitos, CA
Permit 1398