Privacy-Preserving Secure Contact Tracing

ConTraIL* Project:

P. Madhusudan*, Ling Ren*, and V.N. Venkatakrishnan**

*University of Illinois at Urbana-Champaign
**University of Illinois at Chicago / Discovery Partners Institute

May 8, 2020

1 Introduction

We present a general secure and privacy-preserving architecture that articulates a design for contact tracing, and several concrete instantiations of it that realize different privacy policies.

We follow a top-down design principle: from privacy policies and specifications, to a general decentralized security architecture, and then to concrete cryptographic protocols. The first two are also articulated and written in a way that is accessible to people who do not know (or care to go into) cryptography. Furthermore, the first protocol and the second protocol (the main one that we recommend) that we propose require only knowledge of simple cryptographic hash functions, while the third involves more complex cryptographic concepts.

We start in this section with a general overview of manual and digital contact tracing. In Section 2, we concentrate on privacy, and articulate the desired privacy policy for digital contact tracing at a high level that is accessible to policy makers. The policy we advocate is called the "Reveal X policy", parameterized by X which is the information that users *warned* by the system get to know about their exposure from diagnosed users. The rest of the policy is extremely tight, demanding that no information about a user, not even the eids they broadcast, is ever revealed to anyone— the healthcare professionals (HCPs), the server, eavesdroppers, attackers, and other users on the system. The information revealed to warned users (X) range from the ephemeral ids they heard that witness their exposure to diagnosed users, or only the *number* of such exposures, or only the coarse location and time where exposure occurred, or, no information at all.

In Section 3 we turn to other *security attacks* (apart from privacy). The attacks we consider are ways to *misuse* the system, giving it false data that makes the system warn many users who should not be warned. These false warnings can render the system useless and fall out of favor with users. We consider two attacks— the first is people reporting falsely that they are positively diagnosed, and the second is the **relay attack**, where attackers can relay ephemeral ids emitted in one location across the nation quickly, and hence create false "contact" between users who are geographically apart. The first attack is relatively easy to prevent by allowing users to be certified they are positively diagnosed before allowing them to declare positive to the system. However, the latter is much harder to prevent. Furthermore, the second attack can be done silently (i.e., we will not even get to know, even retrospectively, that such an attack has happened), and is *anonymous* (we will not know who did it).

We propose a mechanism that we call *context signature verification* (CSV) that solves long-range relay attacks effectively. We then propose an architecture that is essentially a decentralized architecture that supports CSV. In this architecture, all users keep their personal information on their phone, and any information sent to others is *encrypted* in a fashion that no one else can decrypt. Consequently, all computations and communication happen only on encrypted undecryptable data.

*CONtact TRAcing ILlinois Project: https://github.com/ConTraILProtocols

The protocols we propose using context signature verification prevent relay attacks; and as far as we know, other proposals in the current literature do not prevent them [2, 3, 1].

In Section 4, we turn to concrete protocols that realize our proposed security architecture for various privacy policies. The first protocol does not do context-signature verification, and is roughly similar to protocols proposed by other research groups.

The second protocol, which is the main one we recommend, does context-signature verification (and hence staves off long-range relay attacks). Moreover, in this protocol undiagnosed users keep their information on their phones, not even sending them in encrypted form to others, and *infer* whether they are warned by diagnosed users locally using their phones. The server plays only a *communication conduit* from diagnosed users to undiagnosed ones to facilitate warnings. We propose a third protocol, which also does context signature verification, but reveals less to warned users— warned users do not even get to know the ephemeral ids that witness their exposure to diagnosed users, but only get to know the *number of such contacts*. This protocol (and others that require more coarse information sent to warned users) involve more advanced cryptography (homomorphic computations, multiparty computations, etc.)

Manual and Digital Contact Tracing

The idea of digital contact tracing is simple— the phones we carry can help us determine the set of contacts (close physical distance for some reasonable time determined typically by observing Bluetooth ids) we make with people. When a person tests positive, they can warn their contacts either announcing their id (and letting others figure out if they have been in contact) or pull out their contacts from the last n days (where n is chosen to be the probable time they were contagious). People so warned can self-quarantine themselves, even from their families, get tested for the virus, monitor themselves, and return to society either when they test negative or have overcome the infection.

The primary question we explore in this article is how our phones track contacts and how we compute the contacts to be warned when a person tests positive in a way that (a) preserves the privacy of all users and (b) prevents misuse by malicious users or external attackers.

Our goals of privacy are extremely high— not only do we want interactions to be HIPAA compliant but also that users are protected from the app/server that computes contacts, the healthcare providers, the government, and also external attackers. Our scheme will use cryptography heavily to ensure privacy, similar to the ones used in secure communication on the internet, such as for bank transactions and financial trading.

Manual contact tracing: Let us first remind ourselves about what happens during manual contact tracing[1], since manual contact tracing efforts have been accepted and will be an important component for states to tackle COVID. Our general privacy goals will be *stronger* than what manual contact tracing offers, which is why we think it will be acceptable to users (provided they understand what privacy is preserved). When a user P gets diagnosed as positive (P is sometimes called called Patient-0), epidemiology-aware "disease detectives" conduct interviews with P to determine the set of people P may have had contact with, the places P traveled, ate, slept, etc. and also consult various data sources (reservation logs, airplane seating, etc.). They compile a list of contacts that have been potentially exposed to P. Then a set of people cold-call these contacts to warn them of exposure, and continue by monitoring them, and advising them regarding precautions, quarantine, and tests.

Digital contact tracing is a way of achieving very much the same ends as above without involving humans 'in the loop', but by requiring humans to use an app. In support of manual tracing, we first note that when human experts are involved in deciding the risk and exposure it is actually a plus, as automatic methods lack such human oversight. However, requiring the involvement of humans in tracing every contact is cumbersome, and can also be error-prone as the patient's memory fades. Digital contact tracing is automatic and can be more effective in identifying contacts sometimes (e.g., if P was right next to another buyer in a shopping checkout lane, P may not remember or even know the identity of this other person and manual tracing will miss this contact while digital tracing won't). In sum, we think of digital contact tracing as a complementary and relatively inexpensive way to augment manual contact tracing efforts.

[1] https://www.npr.org/sections/health-shots/2020/03/10/814129534/how-the-painstaking-work-of-contact-tracing-can-slow-the-spread-of-an-outbreak

We want to emphasize that in the context of manual contact tracing, diagnosed users do give up some information regarding their location history and contacts. Their willingness to do so stems from the fact that they want to help in containing and suppressing the virus from spreading. This motivation will likely persist in digital contact tracing as well, as long as their privacy, especially in terms of their identity being not revealed, is preserved.

2 Privacy Policies for Digital Contact Tracing

Our goal of digital contact tracing (DCT) is to preserve users' privacy *even more* than is preserved with manual contact tracing. Systems and people involved in digital contact tracing won't even know the precise contacts made and warned, as they will be anonymized. Computations required for identifying contacts will be done on *encrypted data, without decrypting them* (using modern cryptographic methods). Except the minimal information contained in the warning given to users, no other information will be revealed.

Digital Contact Tracing: Setup
First let us fix some terminology.

- *Users* are people who have joined the digital tracing system, installing an app, and some form of a password protected account.
- *Diagnosed Users* are users who have tested positive for the virus and want to declare themselves positive to the system.
- *Undiagnosed Users* are users who aren't diagnosed users. These are the people who can be potentially warned by the system.
- The *digital contact tracing (DCT) server (or just server)* is the main central system that facilitates communication and digital contact tracing. It is co-designed with apps that execute on edge devices (phones) that are owned by users.

The setup we assume is that users download the contact tracing app on their phones, and set up a unique userid with the central server. Furthermore, they generate locally a secret key, randomly (they are not shared with anyone, not even the server). Ephemeral ids are generated as a function of this secret key and any information it gets from sensors that the phone may have from its environment, including sources of randomness. We assume that these ephemeral ids change frequently with time (this frequency is decided based on other considerations such as whether others can track the phone based on the ids, etc., but are not relevant to the protocol design).

We make two additional restrictive assumptions. First, we assume that we do not have any control on the random ephemeral ids generated. This is in part to conform to the strict Bluetooth emitting schemes proposed by Apple and Google on their phone platforms. Second, we assume that phones cannot do handshake or true communication when they come in proximity; rather, they can only broadcast their id and listen to ids broadcasted. This is again to conform to the Apple-Google methodology; furthermore, there is potential loss of connections if we do handshakes and also possibly more battery drainage. These two restrictions exclude several simpler protocols that can achieve our goals; we do not discuss them in this version of the paper.

Every user broadcasts its ephemeral id. It records the ephemeral ids it has broadcasted locally on the phone, encrypted using a personal key. Every user also listens to ephemeral ids broadcasted by devices around them, and records these ids as well locally on the phone, encrypted using a personal key. The information recorded by each user A are two sets: the tell-set T_A corresponding to broadcasted ids and the heard-set H_A corresponding to ids that it has heard.

When a user B tests positive, the digital contact tracing system's goal is to have users who have heard an id broadcast by B (i.e., those whose heard set intersects with the tell set of B) to get warned. When people test positive, we refer to this set of undiagnosed users as the *warned users*.

2.1 Privacy Policies

The broad privacy policies that our DCT schemes ensure will meet the following ideals.

- First, undiagnosed users in the system will not divulge any information, to anyone (to diagnosed users, to other undiagnosed users, to the central servers, to health officials involved in DCT, to external attackers, etc.).
- When a user P tests positive, the digital contact tracing (DCT server) will not know the locations and movements of P. No external attacker of the system will get to know the identity of P, the locations of P, or the identities of P's contacts. All of P's information (ephemeral ids it has broadcasted, etc.) that is provided will be *encrypted* before being sent to the server and stay encrypted forever.
- Undiagnosed users who do not get warned by P will get to know absolutely no information, except the 1-bit information that they are not in the set of users being warned. Hence, these users won't even know the users P warns through the DCT system, nor know anything about P, like P's identity, its ephemeral ids, etc.

Given the above, the only thing that remains is to decide what information flows from the diagnosed user P to the users being warned, i.e., those users who have heard an id broadcasted by P. These users will, of course, will get to know, at the least, that they have been in contact with someone who has tested positive. But what other information do they get?

We envisage several kinds of information warned users can get, and hence parameterize the privacy policy by this information revealed, called X (for eXposure). Note that here is a bit of privacy lost here— warned users may get to *infer* who P may be using this information X (at least a bit more than without knowing X). However, the information serves a utility as well, as this information can help undiagnosed users gauge their level of risk better.

Information revealed (X) can be one of several choices. First, we can reveal the ephemeral ids that witness the contact with diagnosed users. In this case, since each user can keep track of where and when they heard these ids, they would know the (approximate) time and place where their exposure to the diagnosed patient took place. The second possibility is to reveal the *number* of distinct eids that they heard that belonged to the diagnosed patients, but not reveal the eids themselves. This will protect the privacy of the diagnosed patient more and yet give some information on the degree of exposure of the user to diagnosed users. The third possibility is to not reveal the eids of contact but reveal a coarse window of times and coarse locations where the user was exposed.

We now formalize this privacy policy parameterized by X, by specifying the precise information flow between various parties in the system.

Reveal X policy: In this policy, the warned users get to know that they are warned and in addition, get to know their exposure in terms of the information X.

Reveal X policy: Information flow:

From	To	Information revealed
Undiagnosed users	All other users, servers, attackers of servers	None
Positive patient P	Servers, attackers of servers	None
Positive patient P	Users not warned by P	The fact they are not warned
Positive patient P	Users warned by P	The fact that they are warned and the information X associated with the exposure

In the above, X can be one of the following;

- Contact eids: Reveal to warned user the ephemeral ids that they heard that corresponds to the exposure to the diagnosed individual.
- Number of eids: Reveal to warned user the number of ephemeral ids that they heard that corresponds to the exposure, but not the eids themselves.

- Coarse location and time: Reveal to warned user nothing regarding eids, but just the coarse locations and times of exposure (the granularity of this coarseness can be fixed appropriately).
- Nothing: Reveal to warned users nothing, except that they have come in contact with a diagnosed user.

Note on the policies: As we mentioned earlier, revealing contact eids to the warned user that marks their exposure reveals, essentially, the rough locations and times as well, as users can track these themselves. The idea of the "Reveal coarse location and time" policy is to not reveal eids and reveal location and time at a level of coarseness that can be tuned (for example, reveal the two-hour periods of exposure).

Note that even in the "Reveal nothing policy", it is possible that a warned user can infer the identity of the diagnosed user (say if the warned user only had the single contact meeting for the entire period). Indeed, the very fact that we are doing contact tracing reveals this information. In this white paper, we consider inference attacks only based on what is exchanged through our mechanisms; attacks that combine "out of band" information are considered mostly out of scope.

The main techniques that facilitates building contact tracing that preserves the above privacy properties rely on *computation on encrypted data*. In particular, the techniques in modern cryptography, ranging from cryptographic hashes to private set intersection [5] and *fully homomorphic encryption* [4]. We will not dwell on them now (see Section 4 where we discuss them); the choice on the actual cryptographic mechanisms will depend on the privacy policy, performance of cryptographic computation, and communication bandwidth concerns.

Our approach is to *separate* the privacy policy from the *mechanism* to achieve it. At the higher level, people who decide on the policy need to understand that the agents can communicate and do computation on encrypted data all the time, without having to to decrypt it, and hence realize these policies.

High-level protocol specification: One way to think about the policies above is that computation and communication just happen magically on encrypted data. When a user P tests positive and sends its ephemeral ids, etc. to the server, it's encrypted. The server can work with users to compute whether they should be warned or not and the information that needs to be conveyed while keeping all communication and computation on encrypted data.

In fact, our mechanisms can be thought of as following *high-level specification* where no information is revealed other than the specification say, even to external attackers:

High-level specification for Reveal X policy:

- Every user A broadcasts a set of ephemeral ids (possibly generated from a long-term secret key k_A, based on time and a random source, that is not known even to the server).
- Whenever A and B are in close proximity (they are in closeby locations at the same time), they can possibly record each other's ephemeral ids. It is not guaranteed that if A observes B's id, then B would definitely observe A's id too.
- When user B gets diagnosed positive, B has the option of declaring a (possibly redacted) set of ephemeral ids broadcast by them. We want every undiagnosed user A who has heard at least one of B's declared ephemeral ids to be warned. The warned user only gets to know that they have been warned ant the information specified by X.
- The digital contact tracing system satisfies the Reveal X privacy policy. In particular, undiagnosed users who do not get warned should not get to know anything except that they are not warned. The healthcare providers in the system, the servers, any attackers who snoop in on the servers' computation or communication, should not get any information whatsoever about diagnosed or undiagnosed users (their identity, the ids they broadcasted or heard, etc.).

3 Security concerns and architecture design

Now we turn our attention from privacy to other security concerns. While privacy can be ensured using computation on encrypted data, we also need to make sure the system is not *misused* or subject to attacks that render it useless. In fact, these security considerations are the primary ones that determine the architecture we propose. We first define a security threat model.

Security Threat Model

On the server side, we expect the server to be running the protocols as described later in the section. Therefore, we do expect the server platform to be well protected by state-of-art defenses against software attacks from malicious third parties, so that it does not deviate from the published functionality, and also for protection of the data that it stores from tamper. Note that, according to the previous section, the server does not learn any sensitive information about diagnosed or undiagnosed users. We assume that attackers may observe the computation of the server and observe communication between the server and other devices. The server is assumed hence to be *honest-but-curious*.

On the edge device side, we expect a vast majority of users to run the published protocol as described. However, we do expect a small set of malicious users who might use a modified app (or platform) to supply the server with bogus data.

We also do not directly address denial-of-service attacks caused by uploading false meeting information from a large number of edge devices (though these can be mitigated through the use of some proof-of-work or captcha schemes). We also do not address *side-channel* attacks on the server, and assume the server is protected from such attacks by other means.

Security Attacks

Our protection schemes are designed to ensure that such malicious users do not undermine the security of the system. Mechanisms to prevent these attacks make our architecture different from others proposed in the literature so far. These attacks try to cause *false warnings*— warnings to people who never came in contact with a positively tested person, which can eventually cause the system to be mistrusted and ignored by people.

False reporting of infections by individuals: One attack to consider is a malicious user M, who goes close to certain people of interest, say important people, leading them to record his ids. The user M can then declare themselves positive to the system resulting in several people being warned of infection unnecessarily.

Relay attacks: The second attack we consider tries to increase false warnings by using communication (say over the Internet) to perform relay attacks. An attacker could take ids that are broadcast by phones in one vicinity R and relay them in another place S at either the same time or at another time. The receivers at R and the broadcasters at S could be phones or even Bluetooth receivers/beacons placed at strategic locations with high traffic. When a person who is near R tests positive and reports their ids to the system, this can lead to false warnings for people at S. Notice that these attacks are also *silent*, especially in a privacy-preserving system— we wouldn't even know that it is happening. Users who get warned are unlinkable to the people who warned them, and therefore cannot be correlated after they get warned. Large sets of people can get warned falsely by such an attack and we wouldn't even know at the time of attack or even retrospectively.

Mitigation for false reporting and relay attacks:

Solution for false reporting of infections using health official confirmation of positive diagnosis: The solution to mitigate false warnings by individuals (and this is commonly proposed in most schemes in related proposals) is to have a 'human in the loop' that is involved in certification of people in order to affirm that an individual has been tested positive. In other words, we cannot have users self-declare themselves with a push of a button that they are positive. Rather, just as in manual contact tracing, we would need

a health official to attest that the person is indeed positive and that their contacts can be warned. Note that a malicious user who also happens to test positive can still mount such an attack— however, this does not result in large scale attacks.Note that such attacks can also happen in manual contact tracing. We can hence safely assume, for the large part, that positively infected users are truly positive and the information they report is genuine.

Solution for broadcast attacks using Context Signature Verification (CSV): Preventing relay attacks at first glance looks extremely hard— how do we prevent an attacker from listening to an id in California and replaying it in Illinois, almost immediately? We are not aware of any protocols published by other groups that prevent this attack.

If BLE based contact tracing afforded phones to truly communicate *interactively* each time they paired, then we can build a solution where the phones exchanged information that encoded the current location using the other's public key, for example, in order to ensure these messages when replicated elsewhere to another user would be meaningless. However, in BLE based contact tracing, there isn't much of true communication (back and forth) between phones. Rather, each user's phone just beams an ephemeral id and listens to ephemeral ids broadcast by other phones nearby and records them. An analogy here is that every person can be seen as walking around with an ephemeral id held up on a placard above their heads and users just observe these ids and record them, rather than stop and talk to each other! This kind of simple broadcast of ids and listening to ids is crucial for practical effectiveness, to be able to observe lots of ids very fast (say when in a crowd) and to not expend battery in communications.

We propose a novel solution here based on what we call *Context Signature Verification* (CSV) to prevent relay attacks. The idea here is to *not* prevent relays when they happen but rather catch them in the end when users are warned. More precisely, each user tracks along with the BLE contact ids a *signature of their context when they observed the id*. For example, a signature of the context could be a cryptographic hash of their location and time, in which case it will be sequence of bits that look random to any observer— it's impossible to derive the original location or time (or even any partial knowledge of them) from the hash.

When a diagnosed user I reports their id that A has heard of, we require I also to report the context signature when that id was broadcast by I. User A would then check whether context signature matches her own signature of the context she recorded when she heard that id, and accept the warning as a real warning only if they do. Context signatures need to be a bit forgiving in accuracy— as long as the locations and times approximately match, A should consider herself warned (see Section 4 for some ways to handle this). Note that if an attacker relays an id from a user A in a location to another (reasonably far) location where there is a person who later tests positive, then when that user warns A, A would see a mismatch of context signatures and disregard it.

Intuitively, a user broadcasting a BLE id at a particular context (location and time), knows this context, and all phones closeby observing the id at that time know the context as well. However, when the same id is relayed to a different place or time, a user who hears them there would hear it in a different context. Consequently, if a diagnosed user and a warned user can verify that the context in which they sent and received the id are the same, we can prevent relay attacks.

3.1 The High-level Secure Privacy-preserving Protocol Architecture

We are now ready to propose the architecture of our high-level protocol aided by the two solutions above— to have health officials certify positively diagnosed users and to support context signature verification. Figure 1 illustrates our architecture.

1. Every user A has a randomly generated secret key k_A. These are held secret even from the server and app platform. They broadcast a sequence of ephemeral ids that change with time, and that depend on k_A (and possibly sources of randomness), using a function $ephid(k_A, t, rand)$.

   This function $ephid()$ must be one-way (hard to invert) and non-malleable— any person who doesn't know k_A should find it computationally impossible to look at the id and determine k_A or compute ephemeral ids for other times. They should also have high enough entropy to prevent brute-force attacks and collisions.[2]

[2]The proposal from Apple and Google for generating ephemeral ids using keys and temporary-exposure-keys satisfy this requirement.

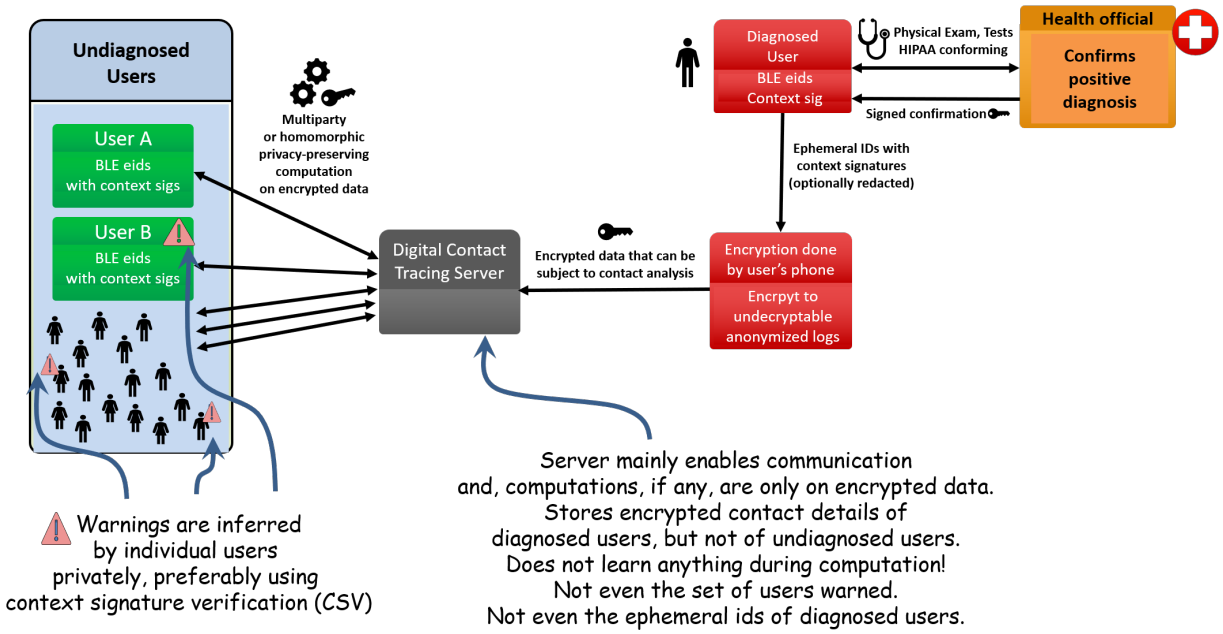

Figure 1: Proposed high-level secure privacy-preserving contact tracing architecture that supports context signature confirmation (CSV)

2. Every user A records the ephemeral ids broadcast by phones nearby, and records them with a cotext signature (a hash of the time and location when each broadcast is heard), to create a heard set set H_A consisting of pairs of the form $(ephid, csig)$.

   Every user A also keeps track of their own ids and the associated context signatures in a tell-set T_A that also has pairs of the form $(ephid, csig)$. $csig$ if there is no context signature verification). Elements in these sets are constantly flushed when they get too old (beyond the contact time for tracing, ~ 2 weeks).

   The context signature is omitted in these sets if the protocol is not doing context signature verification.

3. When user B gets positively diagnosed, they do the following:

   (a) Contact a health professional who can *cryptographically sign* that the user has a positive diagnosis.

   (b) B will encrypt its tell-set and send it to the DCT server.

   (c) All undiagnosed users will interact with the server to find out, privately, whether they are warned or not. This will happen by computation on encrypted data. In particular, users will never give up their information unencrypted to the server. There are two ways to achieve this: either they download data from the server, and privately compute their warning status, or they communicate by sending their data only in encrypted form, and in tandem with the DCT server, compute their warning status. In either case, the server will *not* get to know the warning status of undiagnosed users. Undiagnosed users who do not get warned will not get any further information. Undiagnosed users who do get warned will get the information X prescribed by the privacy policy.

In the above architecture, the server will have no (unencryptable) information of users— not even their ephemeral ids or the warning status of undiagnosed users. When users are warned, it will be up to them to decide what they should do— there will be generic information as to how they can contact a health professional to seek advice regarding quarantine, monitoring, and testing, and later, if they test positive, to become a diagnosed user who declares themselves to the digital tracing system to warn *their* contacts.

The server in the architecture gains no information and communication channels are secured; hence a passive observer of the server, its data, and the communication can also gain no information about users.

Note that we do not specify precisely *how* diagnosed users will encrypt the information sent to the server, or how the the server and undiagnosed users communicate and compute so that users infer their warnings privately. These will depend on the precise Reveal-X policy chosen and whether context signature verification is performed. In Section 4 we will describe several instantiations of this architecture where warning analysis by an undiagnosed user and the DCT server is performed using techniques ranging from simple crypto hashing that facilitates homomorphic *equality* checks on encrypted data, and more complex homomorphic and multiparty computation for private set intersection. Also, we propose in Section 4 concrete schemes for achieving context signature verification using a technique based on *neighborhood gridification*.

4 Concrete Protocols with Cryptographic Schemes

We now examine some concrete protocols for contact tracing. The first protocol (Protocol EID-Hash) is *not* an instance of the architecture that we propose since it does not use context signature verification (CSV), but is the simplest protocol, and similar to proposals that have already been suggested by several other groups. This protocol meets the "Reveal Contact Ephemeral Ids" policy. We discuss it mainly to set a baseline, and show a concrete protocol that is subject to relay attacks.

We then describe the first protocol that we do recommend, Protocol CSV-Hash, that does conform to our architecture, performing context signature verification and preventing relay attacks. This protocol meets the "Reveal Contact Ephemeral Ids" policy as well. This protocol requires new techniques for doing context signature verification, in particular in reducing context signature verification to *equality* of encrypted values, which facilitates simple crypto hash based solutions. These techniques rely on gridification of neighborhoods, and we discuss it in detail in Section 4.2.

The third protocol that we propose, Protoco PSI-CSV, is one based on multiparty computation for computing Private Set Intersection (PSI). The key differences in terms of the policy here is that unwarned users get to know not even *hashed eids*. Furthermore, since no data is published, intersection services can be withdrawn after the time of contagion by a diagnosed patient is over. We recommend a particular RSI based PSI protocol that we believe will have good performance in practice in catering to large populations.

The fourth protocol we propose, Protocol PSICard-CSV, aims to satisfy a stricter privacy policy, namely "Reveal Number of Eids", while at the same time doing context signature verification. The only information a warned user learns is the number of meetings that give warnings (as opposed to which exact meetings give warnings). This makes it hard for a warned user to identify the diagnosed users (assuming the warned user had a reasonable number of meetings). This problem can be solved by implementing any homomorphic or multiparty computation for checking intersection of sets, where when sets have nonempty intersection, only the cardinality of the intersection is revealed. There are again several methods to realize this, and we present a simple one that uses Diffie-Hellman based Private Set Intersection, with permutations. However, we do *not* yet recommend this policy for large populations, as we are not sure it will scale well. Exploring other protocols for PSI-Cardinality are required to find a scheme that is better in required computation and communication.

In this version of the whitepaper, we do not have concrete recommendations for other policies, such as "Reveal Coarse Location and Time" policy or "Reveal Nothing" policies. These seem to require homomorphic or multiparty computation of more complex computation. While these are definitely *possible* and several schemes already do exist, it is not clear to us which mechanisms scale best for digital contact tracing (large number of users, edge devices have low computational power while server has high computational power, communication bandwidth over cellular networks, etc.). We also think it is prudent to wait to see whether these stricter privacy policies are really called for, in consultation with health systems and user opinions, before designing expensive solutions for them.

To summarize, the four protocols we propose in this section are given in the following table, and where we recommend Protocol CSV-Hash and Protocol PSI-CSV only (the latter for better privacy at a slightly more complex implementation cost).

Protocol	Privacy policy (info. to warned users)	Context Signature Verification?	Techniques
Protocol EID-Hash	Reveal Contact EIds	No	Crypto hashes
Protocol CSV-Hash*	Reveal Contact EIds	Yes	Crypto hashes + Gridification
Protocol PSI-CSV*	Reveal Contact EIds	Yes	Crypto hashes + Gridification + Private Set Intersection
Protocol PSICard-CSV	Reveal Number of EIds	Yes	Crypto hashes + Gridification + Private Set Intersection Cardinality

*: Recommended

Ephemeral ID generation

While we do not specify here how ephemeral ids are generated, there are several schemes proposed by other groups. The schemes published by EPFL and Google+Apple recommend schemes based on generating day-keys and then time-specific keys that seem to be motivated by other concerns. First, to reduce the communication from the diagnosed users to the server and the server's communication to other users. Second, there may be a worry that *generic* app manufacturers may not change ephemeral ids often enough, and hence a deterministic scheme enforced by the phone would be useful. In any case, our protocols are compatible with such ephemeral id generation as well, as we do not have any specific requirements on how ephemeral ids are generated (however, in our schemes, we won't recommend giving the key to generate ephemeral ids to anyone).

4.1 Protocol EID-Hash: A simple protocol without CSV (not recommended)

We describe first the simplest protocol we can think of for the Reveal Contact Eids policy. It does not use context signature verification, and is hence subject to relay attacks. Most proposals that we know in the literature resemble this protocol.

The protocol

Let us call a generic diagnosed user Bob (B), and a generic undiagnosed user Alice (A). We fix a (deterministic) cryptographic hash function $Hash$ for the platform that all users know.

When Bob gets diagnosed positive and certified by a health professional, he takes the (possibly redacted) set of ephemeral ids in his tell set T_B the last n days (n determined by contagion probability) and sends their hashes to the server. More precisely, Bob sends to the server the set

$$\{Hash(eid) \mid eid \in T_B\}$$

The server publishes them for any user to download.

An undiagnosed user A downloads this published set P time to time to her phone. She then checks whether the intersection ofhashes of eids in her heard set H_A (in the last n days) and the downloaded set P intersect. I.e., she computes

$$I = \{Hash(eid) \mid eid \in H_A\} \cap P$$

If the above computed set I is empty, A considers herself not warned. If the computed intersection is nonempty, then A considers herself warned, and furthermore, the set of all eid in her heard set H_a such that $Hash(eid) \in I$ are the contact eids of exposure to diagnosed individuals.

Attacks and Privacy

- Protocol is subject to relay attacks.

- Ensures roughly the "Reveal contact ephemeral ids" policy; more precisely the following information flow:

From	To	Information revealed
Undiagnosed users	All other users, servers, attackers of servers	None
Positive patient	Servers, attackers of servers	Hashed eids broadcast
Positive patients	Users who do not get warned	The fact they are not warned and hashed eids
Positive patients	Warned users	The fact that they are warned and the eids associated with the exposure

- Warned users know the meeting ephemeral id contacts that led to the warning, which reveals location and time of contact as well, as undiagnosed users can track when and where they heard these eids. This information can be used to guess more accurately who the diagnosed user(s) could be. On the other hand, knowing this information is useful, as it allows warned users to assess their risk of exposure better.

- Users who are not warned do not gain much information by looking at the published list of hash eids, except that there have been people diagnosed, the total number of meetings they collectively report, and that they haven't been in contact distance with them. In particular, unwarned users do not even get to know the ephemeral ids (though they get to know their hashes).

The above mechanism of hashing the ephemeral ids is similar to several proposals in the literature (in particular, PACT and DP-3T).

4.2 Protocol CSV-Hash: A Protocol for the Reveal Contact EIDs policy with Context Signature Verification (recommended)

We present now the main protocol we recommend which satisfies the Reveal Contact Ephemeral Ids policy and does context signature verification. Our first goal is to reduce checking proximity between contexts to *equality* checks, and hence to set intersection, as in the protocol above. We first describe the primary technique to achieve that.

Context Signatures and Gridification of Neighbors

The idea of a context signature is to capture the context in which an ephemeral id was broadcast and heard. Intuitively, since ephemeral ids are randomly generated and constantly changing, each ephemeral id has a unique small set of locations and times associated with it that the phone that broadcasted it and the phones that heard it *know*. When they later communicate, one warning another, we use the fact that they share this unique information to stave relay attacks.

The simplest solution would be for the broadcaster of an ephemeral id *eid* to take a hash of the set of $\langle x, y, t\rangle$, denoting latitude, longitude, and time, where the eid was broadcast. Of course, the location sensor in phones are not accurate and also the receiver of the id could be a bit further away— so we take a fairly coarse granularity of the location (say 10m to 15m). Also, to reduce the number of signatures we record, we take a coarse granularity of time as well.

One key property that we need to do to reduce *proximity* of space-time co-ordinates to *equality* checking. The reason is that we will take a cryptographic hash of the context, and hence proximity checking of contexts corresponding to two context signatures will be hard. The coarse nature of the context helps in doing this as well. We can think of each context as a grid in space-time.

However, if two phones are near grid boundaries, they may be on different grid points and yet very close. Hence, if a phone is near a grid boundary, it should compute not just the cell it lies in but also the *neighborhood* cell. In particular, at least one of the two phones should look at its locations and time, and if it is close to a grid boundary, compute the cells close to it and take them as contexts

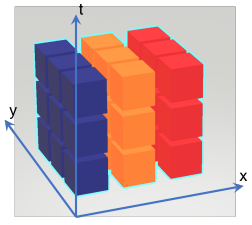

Figure 2: Gridification; 26 neighboring cells

as well. For three dimensions (x, y, t), this means that the phone could add 7 more cells as contexts. We assume we take a crypto hash of these grid points to get the *context signatures*:

$$csig = Hash(x||y||t).$$

Protocol

Every user A broadcasts their ephemeral id and for every ephemeral id also records the context signatures corresponding to the ids. For eids in the tell-set, the user records the context signature corresponding to their cell in 3D, while for the heard-set, the user records the context signature to their cell as well as any neighboring cells that are very close by (at most 7 of them). The tell-set and heard-set now consist of pairs of the form $(id, csig)$.

When Bob gets tested positive, he sends the hashes of his tell set to the server:

$$\{H(eid||csig) \mid (eid, csig) \in T_B\}$$

The server publishes these hashes.

Every undiagnosed user A takes every pair in its heard set $(id, csig) \in H_A$ (which include possibly some neighboring cells) and computes $H(id, csig)$. It then takes the intersection of this set with the published set to determine the exposure. The user considers themselves warned if the intersection is nonempty and not warned otherwise. If warned, it can, as in the previous protocol, find the eids corresponding to the exposure.

Attacks and Privacy

- Protocol is *not* subject to long range relay attacks (beyond the immediate neighborhood of grids) because of context-signature verification.
- Ensures roughly the "Reveal Contact EIDs" policy; more precisely the following information flow:

From	To	Information revealed
Undiagnosed users	All other users, servers, attackers of servers	None
Positive patients	Servers, attackers of servers	Hashed eids||csig broadcast
Positive patients	Users who do not get warned	The fact they are not warned and hashed eid||csig
Positive patients	Warned users	The fact that they are warned and the eids associated with the exposure

- The context signatures are carefully handled in the protocol as they are of low entropy. Hashing them along with ephemeral ids (which have high entropy) guards them from brute-force attacks. Unwarned users and even the server do not have the ephemeral ids of diagnosed users, and hence cannot learn their context signatures. Only truly warned users know the eids of the diagnosed user, but then they know the context as well, anyway.

4.3 Protocol PSI-CSV: A Protocol for the Reveal Contact EIDs policy with CSV using Private Set Intersection

The protocol above publishes the hashes $H(eid||csig)$ of all diagnosed users. Strictly speaking, revealing these hashes to unwarned users does not fully conform with the the policy that unwarned users learn nothing but the fact that they are not warned. This section provides a protocol that uses private set intersection (PSI) to hide these hashes.

The broadcasting and listening steps are identical to Protocol CSV-Hash above. When Bob tests positive, he similarly sends the hashes of his tell set $S_B = \{H(id||csig) \mid \langle id, csig\rangle \in T_B\}$ to the server. But the server does not publish it. Instead, each undiagnosed user Alice will compute her own heard set $S_A = \{H(id||csig) \mid \langle id, csig\rangle \in H_A\}$. Alice and the server will employ a privacy-preserving cryptographic protocol to jointly compute the intersection between S_A and S_B. More generally, S_B will stand for eids of all diagnosed users

in a time window (like a day). At the end of the protocol, Alice learns the elements in the intersection while the server learns nothing.

There are several protocols to achieve multiparty computation of PSI. We propose one variant here based on RSA that we believe will work efficiently in practice.

RSA based PSI

We write $S_A = \{x_1, \cdots, x_n\}$ and $S_B = \{y_1, \cdots, y_m\}$. Note that each element is a hash of some ephemeral id together with a context signature. The RSA-based PSI protocol works as follows.

1. (Setup, once at the beginning) The server picks an RSA key pair $((N, e), d)$, publishes (N, e) visible to all users and keeps d private.
2. For each $y_j \in S_B$, the server computes $y'_j = y_j^d \mod n$ and publishes $H(y'_j)$.
3. For each $x_i \in S_A$, Alice picks a random number $r_i \in \mathbb{Z}_n^*$ and computes $z_i = x_i \cdot r_i^e \mod n$. Alice sends the list $\langle z_1, \ldots z_n \rangle$ to the server.
4. For each z_i received from Alice, the server computes $z'_i = z_i^d \mod n$. The server sends the list $\langle z'_1, \ldots, z'_n \rangle$ to Alice.
5. For each z'_i received from the server, Alice computes $x'_i = z'_i / r_i \mod n$ and $H(x'_i)$. If $H(x'_i)$ matches any of the $H(y'_j)$ published by the server, Alice adds x_i to the intersection.

For correctness, note that $z'_i = (x_i \cdot r_i^e)^d = x_i^d \cdot r_i \mod n$ and hence $x'_i = x_i^d \mod n$. Thus, $H(x'_i)$ and $H(y'_j)$ match if and only if $x_i = y_j$.

The protocol has good efficiency. Note that Step 2 above is performed by the server only once for each diagnosed user's log, and can be reused across queries from many different users. For each query, the amount of computation Alice and the server perform is proportional to $|S_A|$, which is typically small.

Attacks and Privacy

- Protocol is *not* subject to long range relay attacks (beyond the immediate neighborhood of grids) because of context-signature verification.
- Ensures roughly the "Reveal Contact EIDs" policy but more accurately than the Protocol CSV-Hash; more precisely the following information flow:[3]

From	To	Information revealed
Undiagnosed users	All other users, servers, attackers of servers	None
Positive patients	Servers, attackers of servers	Hashed eids||csig broadcast
Positive patients	Users who do not get warned	The fact they are not warned only
Positive patients	Warned users	The fact that they are warned and the eids associated with the exposure

- The protocol has a desirable sunsetting feature. Once enough time elapses after a user is diagnosed, the server will delete the entries corresponding to them, and users can no longer check intersection with these sets (in Protocol EID-Hash and Protocol CSV-Hash, since the logs are published, users can continue to check intersection forever in the future).

Aside: We can strip out context-signature verification in the above protocol to get "Protocol PSI" that retains all features except protection against relay attacks, retaining other positive aspects of the protocol. We do not recommend this, of course.

[3]Note that strictly speaking, the *length* of the log of diagnosed users is also revealed to users who get warned, and the length of the log of undiagnosed users is revealed to the server and its attackers; however, we do not report this as this is too insignifant. Furthermore, we can easily suppress this by publishing logs of fixed length every day, where some of them are entirely fake eid||csig entries, say where *csig* corresponds to signatures of locations that are out of bounds (like latitude > 90).

4.4 Protocol PSICard-CSV: A Protocol for the Reveal Number of EIDs policy with Context Signature Verification

This protocol aims at a more strict privacy policy, Reveal Number of EIds, but not revealing the EIds themselves. A warned user learns only the number of times they had contact with diagnosed users.

The broadcasting and listening steps are identical to previous protocols. When Bob tests positive, he similarly sends the hashes of his tell set $S_B = \{H(id||csig) \mid \langle id, csig\rangle \in T_B\}$ to the server. But the server does not publish it. Instead, each undiagnosed user Alice will compute her own heard set $S_A = \{H(id||csig) \mid \langle id, csig\rangle \in H_A\}$. Alice and the server will employ a privacy-preserving cryptographic protocol to jointly compute the intersection between S_A and S_B. At the end of the protocol, Alice learns the number of elements (but not the precise elements) in the intersection while the server learns nothing.

There are several cryptographic primitives that can realize the above protocol. We are actively evaluating and comparing the efficiency of several candidates, including permuted private set intersection (PSI), homomorphic encryption, and multi-party computation. Below, we use the Diffie-Hellman PSI scheme as an example for its conceptual simplicity.

Diffie-Hellman permuted PSI

We write $S_A = \{x_1, \cdots, x_n\}$ and $S_B = \{y_1, \cdots, y_m\}$. Note that each element is a hash of some ephemeral id together with a context signature. To use Diffie-Hellman, we assume these elements are random members of a cyclic multiplicative group of order q. The protocol works as follows.

1. A picks a uniformly random integer $\alpha \in \mathbb{Z}_q$. A computes $\{x_1^\alpha, x_2^\alpha, \cdots, x_n^\alpha\}$, randomly permutes them, and sends them to the server. Call this list L_1.
2. The server picks a uniformly random integer $\beta \in \mathbb{Z}_q$. The server computes $\{y_1^\beta, x_2^\beta, \cdots, y_m^\beta\}$, randomly permutes them, and sends them to A. Call this list L_2. The server also raises each element in L_1 to the power of β, randomly permutes them, and sends them to A. Call this list L_1'.
3. A raises each elements in L_2 to the power of α. Call the resulting list L_2'. Lastly, A checks the intersection between L_1' and L_2'.

For correctness, observe that for any x_i and any y_j, $x_i^{\alpha\beta}$ will appear in L_1' and $y_j^{\beta\alpha}$ will appear in L_2'. A match occurs if and only if $x_i = y_j$.

Step 2 above can be performed once by the server, and reused across queries from many different users.

Attacks and Privacy

- Protocol is *not* subject to long range relay attacks (beyond the immediate neighborhood of grids) because of context-signature verification.
- The context signatures are hidden from the server and unwarned users as before.
- Ensures the Reveal Number of EIDs policy under the decisional Diffie-Hellman assumption, which states for a group generator g, and uniformly chosen α, β, γ, $(g^\alpha, g^\beta, g^{\alpha\beta})$ is indistinguishable from $(g^\alpha, g^\beta, g^\gamma)$. The server does not learn any information about S_A. Since the server permutes both L_2 and L_1', when there is a match, A cannot tell which two elements resulted in the match.

  Ensures the policy given by the following information flow:

From	To	Information revealed
Undiagnosed users	All other users, servers, attackers of servers	None
Positive patients	Servers, attackers of servers	Hashed eids||csig broadcast
Positive patients	Users who do not get warned	The fact they are not warned only
Positive patients	Warned users	The fact that they are warned and the number of eids associated with the exposure

References

[1] Apple and Google Privacy-Preserving Contact Tracing. https://www.apple.com/covid19/contacttracing. Accessed: 2020-05-05.

[2] Decentralized privacy-preserving proximity tracing. https://github.com/DP-3T/documents/blob/master/DP3TWhitePaper.pdf. Accessed: 2020-05-05.

[3] The MIT PACT Initiative. https://pact.mit.edu/. Accessed: 2020-04-14.

[4] Craig Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, March 2010.

[5] Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *1986 IEEE Symposium on Security and Privacy*, pages 134–134. IEEE, 1986.