Privacy-Preserving Secure Contact Tracing

ConTraIL* Project:

P. Madhusudan[1], Peihan Miao[2], Ling Ren[1], and V.N. Venkatakrishnan[2,3]

[1]University of Illinois at Urbana-Champaign
[2]University of Illinois at Chicago
[3]Discovery Partners Institute

June 22, 2020

Version notes:

- May 5, 2020: Initial version published on Github.
- May 14, 2020: Added an interactive scheme that prevents relay attacks on the spot (without sending any location information, not even cryptographically hashed, to the server).
- June 22, 2020: Significant restructuring of paper to present key ideas modularly. Peihan Miao joins as co-author. Added two new schemes (a) a non-interative scheme that prevents relay attacks on the spot, and (b) a modified Diffie-Hellman PSI algorithm with better scalability.

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.

We advocate that a digital contact tracing scheme should clearly define its information flow and privacy policy, in a way accessible to policy makers and general public. In particular, we will articulate our privacy guarantees with the "Reveal X" policy, where X is the information that users who are *warned* learn about diagnosed users. For example, the information X revealed to warned users may be the meetings that caused the exposure, or the *number* of such meetings. The rest of the privacy policy is extremely tight, ideally demanding that (i) unwarned users do not learn any information, and (ii) no information about an undiagnosed user is revealed to anyone — the server, eavesdroppers, attackers, or other users of the system.

We next turn to *integrity attacks* to the system, which can mislead the system into warning users who should not be warned. These false warnings can render the system unreliable and fall out of favor with users. Some attacks in this camp include undiagnosed users falsely reporting to be diagnosed, and diagnosed users reporting false data to the server. These are relatively easy to prevent by requiring diagnosed users to be certified by health officials and to demonstrate that they are reporting consistent data (data that can only be generated from a certain secret key). A more subtle integrity attack that is hard to prevent is the **relay attack**, where attackers can relay signals (ephemeral IDs) emitted across two far away locations across the nation in real time, and hence create false "contact" between users who are geographically apart. This attack

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

is much harder to prevent and it can be done *silently* (i.e., we will not find out, even retrospectively, that such an attack has happened) and *anonymously* (we will not know who did it).

The main contribution of this paper is a set of mechanisms to prevent relay attacks. As far as we know, at the time of writing, other digital contact tracing proposals in the literature were vulnerable to relay attacks [1, 2, 3]. A concurrent work for relay attack prevention [4] has weaker privacy as it seems incompatible with private set intersection schemes (see below).

The second focus of our work is to identify scalable *private set intersection (PSI)* protocols uniquely applicable to contact tracing. Intuitively, the users diagnosed positive report to the server their *tell* set, capturing what they broadcast in the field, and the server possesses the union of these tell sets. Each undiagnosed user has, locally on their phone, their *heard* set, capturing what they heard in the field. The system checks, for every undiagnosed user, if its heard set intersects with the tell sets at the server. A PSI protocol allows the user to compute this intersection without revealing its own set or the intersection to the server. We also give a PSI Cardinality protocol that reveals only the *cardinality* (size) of the intersection, rather than the intersection.[1]

The primary advantages of PSI are two. First, it keeps the tell sets of diagnosed users private to the server. The tell set of diagnosed users typically contain information they sent out on the field, and an attacker learning the set and having receivers listen to ids in the real world can attempt to *link* the ephemeral ids to identify diagnosed users. Second, these tell sets, if published to public, cannot be erased even after the data becomes stale. Using PSI protocols for tracing allows us to purge stale data from the server.

In summary, the primary question we explore in this article is how to do digital contact tracing in a way that (a) preserves the privacy of all users to the extent possible and (b) prevents misuse (integrity attacks) from malicious users or external attackers. The salient features of our protocols are in the prevention of relay attacks and in scalable private set intersection.

The rest of the paper is organized as follows. The remainder of this section gives a general overview of manual and digital contact tracing. In Section 2, we articulate our threat model and privacy policy, and discuss the integrity attacks we prevent. In Section 3, we give the architecture of contact tracing system. In Section 4 and 5, we present concrete protocols that realize our proposed privacy policy and defend against relay attacks.

1.1 Manual and Digital Contact Tracing

Manual contact tracing. Let us first remind ourselves about what happens during manual contact tracing[2], which has been accepted as an important component for states to tackle COVID-19. When a user P gets diagnosed as positive epidemiologists 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. Epidemiologists then cold-call these people to warn them of exposure, monitor them, and provide them with medical advice regarding precautions, quarantine, and tests.

Digital contact tracing. 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. The idea is simple — the phones we carry can help us determine the set of contacts we make (by observing and recording Bluetooth encountering). When a person tests positive, they can warn their contacts by either announcing their ids (and letting others figure out if they have been in contact) or notifying their contacts from the last n days (where n is the probable time period they were contagious). People so warned can monitor themselves, self-quarantine, get tested for the virus, seek medical advice, and return to society when they test negative or overcome the infection.

[1] At the final stage of preparing this paper, we came across the Epione project which proposed a similar scheme [5].

[2] 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

Comparing manual contact tracing and digital contact tracing, we first note that it is valuable to have human experts evaluate the risk and exposure; digital contact tracing lacks such human oversight. However, requiring the involvement of humans in tracing every contact is cumbersome and hard to scale. Manual contact tracing 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. For example, if P was right next to another buyer in a shopping checkout lane, P may not remember or even know 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 inexpensive way to augment and scale manual contact tracing efforts.

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. Our general privacy goals will be *stronger* than what manual contact tracing offers, which is why we think it will be acceptable to diagnosed users.

2 Privacy and Security in Digital Contact Tracing

2.1 Terminology and Threat Model

First let us fix some terminology and describe the threat model.

- *Users* are people who have joined the digital tracing system, installed the app, and created password-protected accounts.
- *Diagnosed users* are users who have tested positive for the virus and choose to declare themselves positive to the system.
- *Undiagnosed users* are users who are not diagnosed users. These are the people who can be potentially warned by the system.
- The *digital contact tracing server* (or the server for short) is a central server that facilitates communication and digital contact tracing. It is co-designed and communicates with apps that execute on edge devices (phones) owned by users.

Threat model. On the server side, we assume the server is *honest-but-curious*. That is, the sever will run the protocols as prescribed later in the paper but will try to infer as much user information as it can. 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 prescribed protocol. We do not directly address denial-of-service attacks on the server. We do not address *side-channel* attacks on the server. We assume the server is protected from these attacks by other means (e.g., captcha and secure enclaves).

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 other means to supply the server with bogus data.

2.2 Privacy Policies

We *separate* the privacy policies from the *mechanism* to achieve them. The goal of this section is to specify the policies while subsequent sections give concrete protocols to achieve them.

Ideally, except the minimal information contained in the warnings given to users, no other information should be revealed. In more detail,

- Undiagnosed users in the system will not divulge any information to anyone (including all other users, the server, health officials, governments, external attackers, etc.).

- When a diagnosed user P reports data to the server using its account, the server learns that the owner of this account tested positive. Other than the information involved in the account creation, the server does not learn any other information about P, e.g., the real-world identity of P or P's contacts, the locations and movements of P, etc..
- Unwarned users will get to know no information, except the 1-bit information that they are not in the set of users being warned.

Given the above, it remains to discuss what information flows from diagnosed users to the users being warned. These users will, of course, at least get to know 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). This revealed information X represents the privacy loss of the system — warned users may infer a bit more about diagnosed users from X than without knowing X. X can be one of several choices. For example, most proposals in the current literature 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 time and place where their exposure to diagnosed patients took place. A second choice is to reveal the *number* of distinct eids that they heard that belonged to diagnosed patients, but not the eids themselves. Other example privacy policies include revealing course time/location of exposure and revealing nothing beyond the fact that they are exposed.

A more strict privacy policy protects the privacy of diagnosed users better. But we also remark that a more strict privacy policy is not necessarily a better design because (i) more revealed information does provide more utility as it can help undiagnosed users gauge their level of risks better, and (ii) it may require more expensive cryptographic techniques to achieve a more strict privacy policy. Also note that even in the last and most strict "Reveal nothing" policy, it is possible that a warned user can identify a diagnosed user, say if the warned user only had a single contact meeting for the entire period. This is because in this case, the very fact that the user is warned reveals this information.

The following table formally specifies the precise ideal information flow between various entities in the system under a Reveal X privacy policy. In a practical system like the ones we propose in Section 4 and 5, some extra information flows in the system beyond the ideal policy. For example, each user (diagnosed, undiagnosed, warned or unwarned) divulges some information to nearby contacts. The server, being the facilitator for tracing, learns some limited information about the patients. We will articulate these information leakages in respective protocols.

From	To	Information revealed
Undiagnosed users	Server, all other users	None
Positive patients	Server	None
Positive patients	Unwarned users	The fact that they are not warned
Positive patients	Warned users	The fact that they are warned and the information X associated with the exposure

Table 1: Ideal information flow of the Reveal X policy.

2.3 Integrity Attacks and Mitigations

Now we turn our attention from privacy to other security considerations. We need to make sure the system is not subject to attacks that render its warnings unreliable.

We first mention a class of integrity attacks that are important but outside the scope of this paper: attacks that try to prevent the system from warning at-risk users. These attacks generally fall into the category of denial-of-service attacks as they need to block the BLE connections between users' phones, or block users' Internet connections with the server. Since denial-of-service attacks have to be addressed by other means (orthogonal to system architecture), we do not discuss them in this paper.

Instead, our focus is to design protections against a class of attacks that try to cause *false warnings* — warnings to people who never came in contact with a positively tested patients, which can eventually cause the system to be mistrusted and ignored by people. In fact, these considerations about integrity are the primary ones that determine our system architecture.

False reporting of infections. One attack to consider is a malicious user M, who goes close to other people, leading them to record his ids. The user M then declares himself positive to the system resulting in his contacts to be warned unnecessarily.

Solution to false reporting of infections. The natural solution to mitigate false reporting (and this is commonly proposed) is to have a "human in the loop". 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 indeed tested positive and that their contacts can be warned. The health official does not have to vet the patient's entire report to the server. It is sufficient to attest (i.e., digitally sign) a master random seed that is used to generate all the ephemeral ids of the patient.

This solution will also mitigates a similar attack where a malicious user M gives its broadcasted ids to a truly positive patient P and asks/bribes P to upload these contacts to the server. This attack will be detected at the server because M's ephemeral ids are not generated from P's master random seed.

Note that a malicious user can still willingly contract the virus to mount such an attack. (Or M can convince P to use the same master random seed as M, in which case we consider P and M to be the same malicious user.) However, this does not result in large-scale attacks. Furthermore, such attacks can also happen in a manual contact tracing or an ideal contact tracing scheme. We hence safely assume that with the aforementioned countermeasure, information reported by positive users are genuine for the large part.

Relay attacks. The second attack we consider tries to increase false warnings by relaying communication across locations. 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 user who has been near R tests positive and reports to the server, this can lead to false warnings for people at S. Notice that these attacks are *silent* — we would not detect it is happening at the time of of attack or retrospectively. This is especially true in a privacy-preserving system: users who get warned are unlinkable to the people who warned them from the server's perspective, and therefore cannot be correlated after they get warned.

Solution to relay attacks. Preventing relay attacks at first glance looks extremely hard — if a California user Alice heard the ephemeral id of an Illinois user Bob due to attacker relaying, how can Alice distinguish Bob from another local California user? One soon realizes that the two users must exchange some information about their respective locations to thwart relay attacks. We call such a step *context verification* where the context refers to the time and location at which a user broadcasts and receives an ephemeral id. We will give two mechanisms for context verification in Section 4 and 5.

The first one performs context verification *on the spot*, i.e., when two users meet. In this scheme, each ephemeral id is an ephemeral *verification key* (evk) of a digital signature scheme. Each user broadcasts (evk, C, σ) where C is the user's current context (i.e. time and location), and σ is a digital signature on C produced using the signing key corresponding to evk. A recipient of the above message, Alice, can verify that C is close to her own context, and that σ is a valid signature of it. Only then, Alice records evk in her heard set. Later, if the owner of evk tests positive, evk is reported to the server. Alice will get warned by the privacy-preserving tracing mechanism.

Note that if we want users to perform location checks on the spot, it is inherent that they must broadcast some extra information about their locations alongside ephemeral ids. Sometimes, broadcasting such extra information is undesirable, due to considerations on battery consumption or interoperability with other apps.

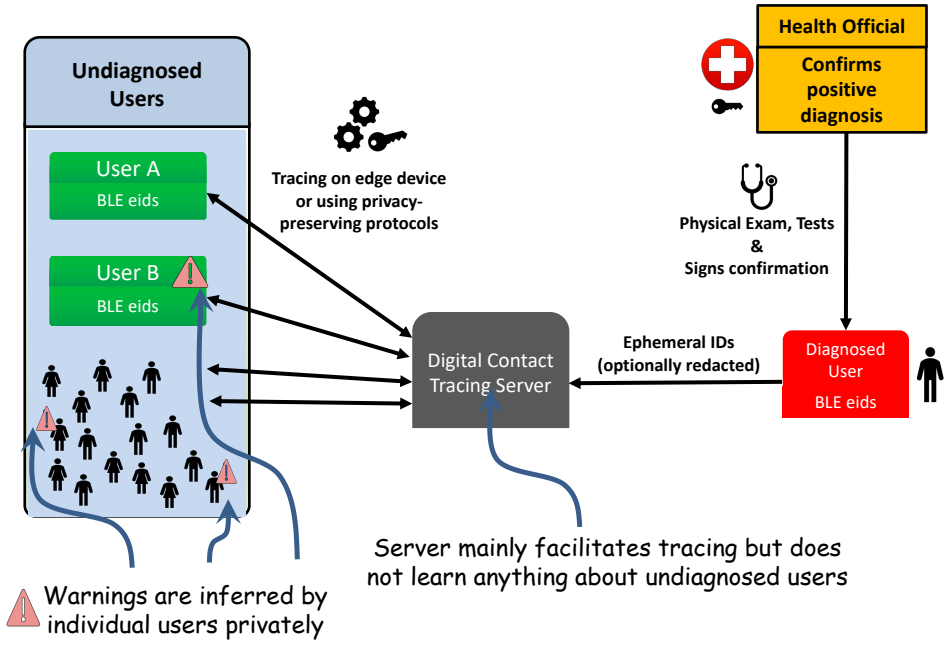

Figure 1: Architecture of secure and privacy-preserving digital contact tracing.

Hence, we give a second solution that still broadcasts ephemeral ids only during meetings and delays context verification in the end when users are warned. More precisely, each user tracks along with each ephemeral id the context in which she broadcasts or receives the id. When a user Bob tests positive, for each ephermeral id he broadcasted, he uploads a hash of the ephemeral id and the associated context. Another user Alice would consider herself warned only if she received Bob's broadcasted id *in the same context* as Bob broadcasted the id. If an id was relayed to Alice from a different place or time by the attacker, Alice would hear it in a different context, and would not consider herself warned.

We remark that in both on-spot or delayed context verification, because smart phone clocks and location sensors can be inaccurate, we have to allow relay attacks within close range, i.e., up to some difference in location and time. Our solutions prevent relay attacks beyond the specified bounded scale.

3 Architecture

We now present the architecture of our secure and privacy-preserving digital contact tracing in Figure 1 and below. The system is broken up into four components: eid generation, meeting, reporting, and tracing.

1. **EID Generation.** Every user A has a randomly generated master secret seed R_A. The user generates a sequence of ephemeral ids (eids) that change every time epoch (e.g., 15 minutes). The eids are derived from R_A and possibly additional randomness and time.

2. **Meeting**. Every user A broadcasts its current eid and records it in a tell set T_A. A also listens to eids broadcast by phones nearby and records them in a heard set H_A. If on-spot context verification is used, a received eid is added to H_A only if it carries a context (time and location) close to the recipient's context. If delayed context verification is used, each recorded eid (in both T_A and H_A) is tagged with the context in which they are broadcast or received. Records in both T_A and H_A are flushed when they become too old, e.g., 21 days).

3. **Reporting.** When a user B gets positively diagnosed, he will contact a health professional who can cryptographically sign his master random seed R_B, which attests that B has a positive diagnosis. B will then report his tell set T_B to the server containing the eids he broadcasted in the recent past (e.g., 21 days), possibly along with the contexts associated with those eids.

4. **Tracing.** All undiagnosed users interact with the server to find out whether they are warned or not. A user A is warned if her heard set H_A intersects with the tell set T_B of some patient B. The tracing step preserves the privacy of undiagnosed users because it either happens entirely on undiagnosed users' edge devices, or uses a cryptographic protocol operating on encrypted data. 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.

When users are warned, it will be up to them to decide what they should do. They can contact a health professional to seek advice regarding quarantine, monitoring, and testing, and later, if they test positive, they can report to the digital tracing system to warn *their* contacts.

3.1 A Baseline Protocol

To put the four components into context, we give a simple baseline protocol for digital contact tracing. Most digital contact tracing proposals in the current literature resemble this protocol. We will also briefly mention the drawbacks of the simple baseline as well as our improvements. The next two sections will present in detail our recommended protocols offering different trade-offs in terms of privacy policies, security guarantees and other considerations.

EID generation. Every user generates locally ephemeral ids pseudo-randomly. The ephemeral ids change frequently (e.g., 15 minutes) to avoid tracking.

Meeting. Every user broadcasts its current ephemeral id, e.g., using Bluetooth Low Energy (BLE). Every user records the ephemeral ids it has broadcasted. Every user also listens to ephemeral ids broadcasted by devices around them, and records these ids as well. The information recorded by each user A are two sets: the tell set T_A corresponding to ids it has broadcasted and the heard set H_A corresponding to ids that it has heard.

Reporting. When a user B tests positive, it uploads its tell set T_B to the server.

Tracing. The system publishes the union of tell sets reported all patients. If a user's heard set intersects with the published tell sets, the user should be warned.

Security analysis of the baseline protocol. The eid generation schemes in the literature are either fully deterministic (coupling all eids in a period) or fully decoupled. Both designs have limitations. Apple-Google [1] adopts a fully deterministic eid generation scheme within each day to reduce the the communication between the server and users. This design creates linkability concerns and does not allow diagnosed users to redact their broadcasted eids. Other designs generate eids in a completely decoupled fashion, leaving the possibility of false reporting attacks.

As we mentioned earlier, a simple exchange of eids is vulnerable to relay attacks. Publishing eids for tracing also has considerable privacy loss, as summarized in Table 2. In particular, it reveals eids of all patient ids to the public including all users.

From	To	Information revealed
All users	Users nearby	Eids
Undiagnosed users	Server, users not nearby	None
Positive patients	Server, all users	Eids of all patients

Table 2: The information flow of the baseline scheme Protocol-EID which publishes patient eids.

3.2 Overview of Our Protocols

To address the limitations of the baseline protocol, we propose several enhancements. First, we will use an eid generation scheme based on both long-term randomness and ephemeral randomness to simultaneously remove linkability and prevent false reporting attacks. Second, we will devise context verification mechanisms prevent relay attacks. We present three schemes: NIOCV (Non-Interactive On-spot Context Verification), IOCV (Interactive On-spot Context Verification), and DCV (delayed context verification scheme). Thirdly, we use PSI and PSI-Cardinality (PSI-Card) to improve privacy.

All our proposed schemes can be composed. We recommend IOCV-PSI and IOCV-PSI-Card as described in detail in Section 4, which use interactive on-spot context verification during meeting and using PSI/PSI-Card for tracing. If there is a constraint in the number of packets that can be broadcast during meeting time, we recommend DCV-PSI and DCV-PSI-Card as described in detail in Section 5, i.e., using delayed context verification instead.

4 Protocols with On-spot Context Verification

4.1 EID Generation

Each user A, upon installing the app, generates a master random seed R_A. It generates a sequence of ephemeral public/private key pairs, one per epoch, (epk_j, esk_j) where $j = 1, 2, 3, \ldots$ is the epoch number. Each ephemeral key pair is generated from the master random seed R_A along with some per-epoch randomness r_j, i.e.,

$$(epk_j, esk_j) = \mathsf{Gen}(R_A, r_j).$$

The ephemeral public key (epk) also serves as an ephemeral id (eid). We use the two terms interchangeably. Depending on the concrete protocol in the meeting step, the ephemeral public key is used either for a Diffie-Hellman key exchange or digital signature verification. In either case, Gen would be a key generation function of a public-key cryptosystem (e.g., elliptic curve).

Looking ahead, when a patient Bob reports epks to the server, he reveals to the server his master random seed as well as all his epoch random seeds. This proves to the server that all the epks are generated from Bob's master random seed. These epks are linkable to the server but are unlinkable to the undiagonsed users (even if we let them download these epks in plaintexts).

4.2 Meeting

The basic idea of a context verification is to capture the context in which an ephemeral id was broadcast and heard. On a genuine meeting (not relayed or replayed), the two users are in close proximity and at the same time, and they both *know* their location and time. We use the fact that they share this unique information to stave relay attacks.

Non-interactive On-spot Context Verification (NIOCV). Following conventions, we write the ephemeral key pair (epk, esk) as (vk, sk) where vk is a public verification key a digital signature scheme and sk is its corresponding secret signing key. Each user broadcasts the following tuple and records vk in its tell set:

$$vk \parallel C \parallel \sigma,$$

where $C = (x, y, t)$ is the context in which the eid is broadcast (x, y, and t denote the latitude, longitude, and time), and σ is a digital signature of C, namely $\sigma \leftarrow \mathsf{Sign}(sk, C)$. If another user Alice receives this broadcast message, she will verify that C is close to her own context and that σ is a valid signature on C, namely $\mathsf{Verify}(vk, C, \sigma) = 1$. If both are true, then Alice records vk in her heard set.

If the attacker performs a relay attack using the same ($vk \parallel C \parallel \sigma$) tuple, then the context would not be close to the recipient. If the attacker changes the context to be close to the recipient and forges a valid signature for it, then it will break the unforgeability of the underlying digital signature scheme. Hence the recipient would not record vk in her heard set.

Saving a packet by gridification of contexts. As an alternative non-interactive scheme, we can allow users to broadcast fewer BLE packets on a meeting. In this scheme, the broadcaster does not broadcast its context; instead, the recipient will verify the signature on her own (nearby) contexts. In particular, each user broadcasts the following pair and records vk in its tell set:

$$vk \parallel \sigma,$$

where σ is a digital signature of its context $C = (x, y, t)$, namely $\sigma \leftarrow \mathsf{Sign}(sk, C)$.

If another user Alice receives this broadcast message, she will verify that σ is a valid signature on her own context C_a, namely $\mathsf{Verify}(vk, C_a, \sigma) = 1$. The clock and location sensor in phones are not accurate and Alice's context C_a could be a bit further away from C. So we need to take a coarse granularity of the location (say 10m) and time (say 5 minutes). We can thus 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 Alice's context is near a grid boundary, she should additionally try to verify if σ is a valid digital signature for any *neighborhood* cell of C_a as context. For three dimensions (x, y, t), this means up to 7 more cells can be viewed as neighboring contexts. If σ is a valid digital signature for C_a or any neighborhood context, then Alice records vk in her heard set.

This alternative scheme prevents relay attacks by the unforgeability of the underlying digital signature scheme. It allows the broadcaster to broadcast fewer BLE packets but requires more computation for context verification on the recipient side.

Interactive On-spot Context Verification (IOCV). We present another interactive scheme that transmits fewer BLE packets on a meeting. We write the ephemeral public/secret key pair $(epk, esk) = (g^a, a)$ to be used in a Diffie-Hellman key exchange. When Alice with (g^a, a) and Bob with (g^b, b) meet, they (their phones) do the following.

1. Alice broadcasts g^a and adds g^a to her tell set. Bob broadcasts g^b and adds g^b to his tell set.
2. Alice receives g^b and Bob receives g^a. Both compute g^{ab}.
3. Alice and Bob use g^{ab} as a symmetric key (e.g., for AES) to encrypt and send their contexts (location and time) C_a and C_b to each other.
4. If Alice finds C_b close to her own context C_a, Alice adds g^b to her heard set. Bob does the same.

We note that unlike the non-interactive schemes, the number of broadcast messages of a user in the interactive scheme grows with the number of its nearby users because the user broadcasts one encryption per nearby user in Step 3.

4.3 Reporting

When a user Bob tests positive, he sends his tell set T_B to the server. Bob can redact his report, if he wishes, by simply excluding some records from T_B. Bob also sends the server (i) his master random seed R_B which has been cryptographically signed by a health official, and (ii) the epoch random seed for each ephemeral public key Bob reports. The server verifies that each eid is correctly generated from the master random seed R_B and the corresponding epoch random seed.

4.4 Tracing

In the tracing step, an undiagnosed user Alice has her heard set H_A, and the server has a set T, that is the union of tell sets from all newly diagnosed patients. Alice is warned if H_A intersects with T. According to our privacy policy, the tracing step needs to provide the following guarantees. First, no entity should learn any information about Alice. Thus, Alice should never upload any plaintexts to the server. Second, if Alice is not warned, she could not learn any further information. Third, if Alice is warned, she additionally learn information X in the "Reveal X" policy.

The simplest tracing scheme is for Alice to download T and locally check if $H_A \cap T$ is empty, as is done in the baseline protocol in Section 3. The information flow of this approach is given in Table 2. In particular, it leaks eids of all patients to all entities both in and outside the system. We next discuss how to use private set intersection (PSI) or PSI-Card protocols to avoid such leakage.

Private Set Intersection (PSI). We write $H_A = S_A = \{x_1, \cdots, x_n\}$ and $T = S_B = \{y_1, \cdots, y_m\}$. Alice and the server will employ a privacy-preserving protocol to jointly compute the intersection between $S_A \cap S_B$. At the end of the protocol, Alice learns the elements in the intersection while the server learns nothing.

There are several protocols to realize PSI. We use a Diffie-Hellman based protocol here that we believe will work efficiently in practice. To use Diffie-Hellman PSI, we assume each element can be converted to a random member of a cyclic multiplicative group of order q. The protocol works as follows.

1. The server picks a uniformly random integer $\beta \in \mathbb{Z}_q$. The server computes $L_B = \{y_1^\beta, y_2^\beta, \cdots, y_m^\beta\}$ and sends them to Alice.
2. Alice picks a uniformly random integer $\alpha \in \mathbb{Z}_q$. Alice then computes $L_A = \{x_1^\alpha, x_2^\alpha, \cdots, x_n^\alpha\}$ and sends them to the server.
3. The server raises each element in L_A to the power of β and sends them back to Alice. Call this list L'_A.
4. Alice raises each elements in L'_A to the power of $\alpha^{-1} \mod q$. Call the resulting list L''_A. Lastly, Alice computes the intersection between L''_A and L_B to learn the set intersection $S_A \cap S_B$.

Privacy and efficiency analysis. Observe that for any $x_i \in S_A$, if x_i is in the intersection, namely $x_i = y_j$ for some $y_j \in S_B$, then $x_i^{\alpha \cdot \beta \cdot \alpha^{-1}} = x_i^\beta$ will appear in L''_A and y_j^β will appear in L_B, which can be detected by Alice. On the other hand, if x_i is not in the intersection, then x_i^β won't appear in L_B with overwhelming probability. Moreover, Alice does not learn any information about $y_j \notin S_A \cap S_B$ because y_j^β looks indistinguishable from a random group element to Alice. This follows from the Decisional Diffie-Hellman (DDH) assumption in the random oracle model.

Using a PSI protocol hides a patient eid (epk) from users who did not come into contact with the patient (including unwarned users as well as users that are warned but not warned by this patient.) The precise information flow is given in Table 3. With on-spot context verification, each user reveals to nearby users an eid as well as the context (location and time) associated with the eid. But since the recipient is in the same (or close enough) context, we do not list it in the information flow chart.

PSI also gives a desirable sunsetting feature. The server can use a different β for records (from patient tell set) from each day, and stop supporting intersection service once enough time elapses after that day. At that point, users can no longer check intersection with these records. In contrast, if the logs are published (in plain tracing), users can continue to check intersection forever in the future.

The Diffie-Hellman PSI protocol is particularly suitable for the contact tracing scenario, compared to OT-based or FHE-based approaches, because it has better efficiency especially on the server side. In particular, Step 1 is performed only once by the server 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.

From	To	Information revealed
All users	Users nearby	Eids
Undiagnosed users	Server, users not nearby	None
Positive patients	Server	Eids of all patients
Positive patients	Unwarned users	The fact that they are not warned
Positive patients	Warned users	The fact that they are warned and eids of meetings causing exposure

Table 3: The information flow using on-spot context verification and PSI.

Another advantage Diffie-Hellman PSI, as we will see next, is that it can be slightly modified to achieve a more strict privacy policy almost for free.

Private Set Intersection Cardinality (PSI-Card). The Diffie-Hellman PSI protocol can be slightly modified to securely compute only the cardinality of the set intersection. We present the PSI-Card protocol below, where the change is highlighted in **bold**. We write $S_A = \{x_1, \cdots, x_n\}$ and $S_B = \{y_1, \cdots, y_m\}$ and assume each element is a random element of a group of order q as before

1. The server picks a uniformly random integer $\beta \in \mathbb{Z}_q$. The server computes $L_B = \{y_1^\beta, y_2^\beta, \cdots, y_m^\beta\}$ and sends them to Alice.
2. Alice picks a uniformly random integer $\alpha \in \mathbb{Z}_q$. Alice then computes $L_A = \{x_1^\alpha, x_2^\alpha, \cdots, x_n^\alpha\}$ and sends them to the server.
3. The server raises each element in L_A to the power of β, **randomly permutes them**, and sends them back to Alice. Call this list L'_A.
4. Alice raises each elements in L'_A to the power of $\alpha^{-1} \mod q$. Call the resulting list L''_A. Lastly, Alice computes the intersection between L''_A and L_B to learn the set intersection **cardinality** $|S_A \cap S_B|$.

Observe that for each $x_i \in S_A$, x_i^β will appear in L''_A, and it will appear in L_B if and only if x_i is in the intersection, namely $x_i = y_j$ for some $y_j \in S_B$. In addition, L''_A is in a randomly permuted order, hence Alice only learns the cardinality of the set intersection. The information flow using PSI-Card is given in Table 4.

From	To	Information revealed
All users	Users nearby	Eids
Undiagnosed users	Server, users not nearby	None
Positive patients	Server	Eids of all patients
Positive patients	Unwarned users	The fact that they are not warned
Positive patients	Warned users	The fact that they are warned and the number of meetings causing exposure

Table 4: The information flow using on-spot context verification and PSI-Card.

5 Delayed Context Verification

Delayed context verification (DCV) is preferred, if due to some constraint, only an ephemeral id can be transmitted at the time of meeting. In this case, context verification will happen at the tracing step. This section presents a protocol with delayed context verification.

EID Generation. The eid generation is almost identical to Section 4, $eid_j = \mathsf{Gen}(R_A, r_j)$, except that each eid in this case is a pseudo-random string and Gen is a cryptographic hash function.

Meeting. Each user broadcasts its eid together with an additional random string s, i.e.,

$$eid \mid\mid s.$$

The broadcaster (resp. recipient) records eid and a seeded hash of the context in which the eid was broadcasted (resp. received). More precisely, each record in the tell set T_A (resp. heard set H_A) has the form

$$eid \mid\mid \mathsf{hash}(s \mid\mid C),$$

where $C = (x, y, t)$ is the context in which the eid was broadcasted (resp. heard) where x, y, and t denote the latitude, longitude, and time. If the attack mounts a relay attack, the context of the broadcaster and recipient would not match, and the recipient will not be warned.

Similar to the alternative scheme in Section 4, the clock and location sensor in phones are not accurate and also the recipient of the eid could be a bit further away, so again we need the context gridification method with coarse granularity. The recipient of an eid should additionally hash the eid with neighboring contexts (up to 7 for three dimensional grids) and add them to the heard set.

Reporting. Reporting is identical to Section 4 except that each record is in the tell set has two parts: an eid and a hashed context with a random seed. As before, the patient can redact its report and the server will verify that each reported eid is correctly generated.

Tracing. The tracing step is also identical to Section 4, and can choose from plain tracing (not recommended), or tracing using PSI or PSI-Card, for a privacy policy similar to Table 2, 3, and 4, respectively. Note that the context C is carefully handled in the protocol as it has low entropy. Hashing a context C along with a random string s hides it from the server and users who did not come into contact.

Comparison with Piertrzak [4]. Our delayed context verification scheme is similar to a recent proposal by Piertrzak [4] but also has fundamental differences. In a nutshell, one scheme tags each record in the tell/heard set with a (hash-based) MAC $\mathsf{hash}(s \mid\mid C)$, and sends the corresponding MAC key s during a meeting. At a high level, the Piertrzak scheme sends an MAC during a meeting, and tags each record in the tell/heard set with the corresponding MAC key. Another minor difference is that the Piertrzak scheme sends the least significant bits of the context during a meeting, while we use the context gridification method to avoid sending these bits.

The main advantage of our scheme is that it is compatible with PSI-Card in the tracing scheme; we do not see how to extend the Piertrzak scheme to work with PSI-Card. The main advantage of the Piertrzak scheme is that a patient sends less information to the server: one record per epoch instead of one record per meeting. If PSI-Card is not desired, we recommend the Piertrzak scheme augmented with our context gridification method.

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] Krzysztof Pietrzak. Delayed authentication: Preventing replay and relay attacks in private contact tracing. Cryptology ePrint Archive, Report 2020/418, 2020.

[5] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. *arXiv preprint arXiv:2004.13293*, 2020.