Accurate and Efficient Trajectory-based Contact Tracing with Secure Computation and Geo-Indistinguishability

Anonymous Author(s)

Abstract. Contact tracing has been considered as an effective measure to limit the transmission of infectious disease such as COVID-19. Trajectory-based contact tracing compares the trajectories of users with the patients, and allows the tracing of both direct contacts and indirect contacts. Although trajectory data is widely considered as sensitive and personal data, there is limited research on how to securely compare trajectories of users and patients to conduct contact tracing with excellent accuracy, high efficiency, and strong privacy guarantee. Traditional Secure Multiparty Computation (MPC) techniques suffer from prohibitive running time, which prevents their adoption in large cities with millions of users. In this work, we propose a technical framework called ContactGuard to achieve accurate, efficient, and privacy-preserving trajectory-based contact tracing. It improves the efficiency of the MPC-based baseline by selecting only a small subset of locations of users to compare against the locations of the patients, with the assist of Geo-Indistinguishability, a differential privacy notion for Location-based services (LBS) systems. Extensive experiments demonstrate that ContactGuard runs up to 2.6× faster than the MPC baseline, with no sacrifice in terms of the accuracy of contact tracing.

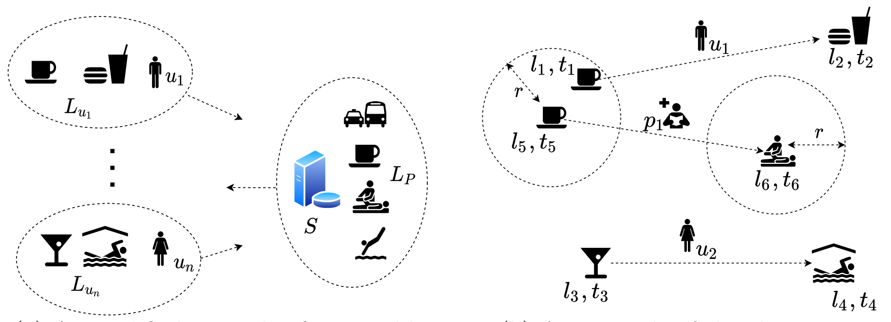
Keywords: Contact tracing · Differential privacy.

1 Introduction

Contact tracing has been considered as one of the key epidemic control measures to limit the transmission of infectious disease such as COVID-19 [13] and Ebola [18]. In contrast to traditional contact tracing conducted normally by interviews and manual tracking, digital contact tracing nowadays rely on mobile devices to track the visited locations of users, and are considered as more accurate, efficient, and scalable [22]. Taking the contact tracing of COVID-19 as an example, more than 46 countries/regions have launched contact tracing applications [18], *e.g.*, TraceTogether in Singapore and LeaveHomeSafe in Hong Kong SAR.

As a complementary technique to the Bluetooth-based contact tracing applications such as the Exposure Notification developed by Apple and Google[1], *trajectory-based contact tracing* [16,9] compares the trajectories of users with the

[1] `https://covid19.apple.com/contacttracing`



(a) A simplified example of our problem. (b) An example of the close-contact.

Fig. 1: (a) A simplified example of our problem. L_{u_i} denotes the visited location of a user u_i. L_P denotes the collection of all visited locations of all patients. (b) An example of the close-contact. u_1 is a close contact, while u_2 is not.

patients, and allows the tracing of both the direct contacts and the indirect contacts. The direct contacts happen when the users and the patients co-visit the same location at the *same* time. In contrast, the indirect contacts normally happen when users and patients visit the same location at *different* times, and the location (*e.g.*, environment, surface of objects) is contaminated and becomes the transmission medium of the virus. Because its capability of tracing both direct and indirect contacts, trajectory-based contact tracing has been widely deployed in countries/regions where there is a stricter policy of COVID-19 control. For example, in China, prior to entering high-risk locations such as restaurants, bars, and gyms, users need to check-in with a health code, which is linked with their personal identities.

Despite the usefulness of trajectory-based contact tracing, the privacy of trajectory data is an obvious issue. In fact, privacy concerns prevent the contact tracing application from being widely adopted in cities like Hong Kong, and there are extreme cases that citizens use two different mobile phones, one for their daily use, and the other one solely for the purpose of fulfilling the check-in requirement when entering the high-risk locations, in order to prevent any privacy breach. Existing privacy-preserving trajectory-based contact tracing research relies on specific hardware (Trusted Hardware such as Intel SGX [16]), but the security guarantee heavily depends on the hardware, and it leads to poor portability. In fact, Intel plans to stop its support for SGX from its 11th and 12th generation processors [15]. Thus, in this work, we aim to develop a hardware-independent software-based solution, which can provide high accuracy of contact tracing, excellent execution efficiency, and strong privacy guarantee.

In this paper, we formulate the *Privacy-Preserving Contact Tracing (PPCT)* problem, which is illustrated with the simplified example in Fig. 1a. We aim to correctly identify whether each user u_i is a close contact or not. If a user u_i co-visited some location with some patient under some spatial and temporal constraints (*e.g.*, co-visit the same location with 5 meters apart, and within a time window of 2 days), then our proposed solution should correctly identify the user u_i as a close contact. The key challenge of PPCT is that there is a strong privacy requirement: during the entire process of contact tracing, the private information (the visited locations of users, together with the timestamps) is strictly protected from the access of other parties.

The challenge of PPCT goes beyond the strong privacy requirement. On one hand, the contact tracing application requires a high level of accuracy. Taking COVID-19 as an example, some countries spend tremendous amount of resources to identify potential close contacts of patients, and sometimes a whole region of citizens need to go through mandated testing after only one case of patient was found in the area. Thus, our proposed solution needs to avoid false negatives as far as possible, maintaining a reasonably high recall, if not close to 100%. On the other hand, we aim to offer an efficient solution to the PPCT problem, as traditional Secure Multiparty Computation (MPC) techniques usually induce unpractical running time. As our experimental results show, a naïve MPC solution requires more than 5 days for 1 million users, which is the population of a modern city. Such long running time prohibits its daily execution, because it could not even terminate within 24 hours.

To tackle the aforementioned challenges, we propose a novel technical framework ContactGuard. ContactGuard improves the efficiency of MPC operations by selecting only small subsets of users' locations to compare with the patients. The subset selection process is assisted by an efficient privacy-preserving mechanism – Geo-Indistinguishability (Geo-I) for Location-based services (LBS) systems. The basic idea is that each user perturbs his/her true visited locations with Geo-I and submits only the perturbed locations to the server, where the patients' locations are stored. Using the perturbed locations of the user, the server compares them with the patients' locations, and then informs the user about which locations are more "risky", because they are closer to the patients. After that, the user could use MPC only on the high-risk locations, to largely reduce the computational overhead.

To summarize, we make the following contributions in this paper:

- We formally define an important problem, the Privacy-Preserving Contact Tracing (PPCT) problem in Sec. 2, which addresses the privacy issue in trajectory-based contact tracing.
- We propose a novel solution ContactGuard in Sec. 3. It combines the strengths of two different privacy-preserving paradigms: differential privacy and secure multiparty computation. The running time is improved significantly because it selects only a small subset of locations to compare with the patients during the MPC operation.
- We conduct extensive experiments to validate the effectiveness and efficiency of ContactGuard in Sec. 4. It runs up to 2.6× faster than the MPC baseline, with no sacrifice in terms of the accuracy of contact tracing. For moderate privacy budget settings for each user, ContactGuard obtains close to 100% recall and 100% precision.

In addition, we review related works in Sec. 5 and conclude in Sec. 6.

2 Problem Definition

In this section, we introduce some basic concepts, the adversary model, and a formal definition of the Privacy-Preserving Contact Tracing (PPCT) problem.

2.1 Basic Concepts

Definition 1 (Location). *A location $l = (x, y)$ represents a 2-dimensional spatial point with the coordinates (x, y) on an Euclidean space.*

Definition 2 (Trajectory). *A trajectory is a set of (location, timestamp) tuples indicating the visited location and time that the location is visited. A trajectory $L = \{(l_1, t_1), \ldots, (l_{|L|}, t_{|L|})\}$, where $|L|$ is the number of visited locations in the trajectory.*

Examples: let location $l_1 = (300, 500)$ and l_2 be another location $l_2 = (200, 200)$. An example of trajectory $L_{example} = \{(l_1,$ 2020-06-10 12:28:46$), (l_2,$ 2020-06-10 17:03:24$)\}$ indicates that location l_1 is visited at timestamp 2021-06-10 12:28:46 and location l_2 is visited at timestamp 2021-06-10 17:03:24. The size of $L_{example}$ is 2, *i.e.*, $|L_{example}| = 2$.

Here, the definition of *trajectory* includes both short sequences (*e.g.*, a user leaves home in the morning, moves to the working place, and comes back home at night) and longer sequences across multiple days. In our problem setting, in order to trace contacts, the trajectory of a user contains the union of all visited locations of a particular user in the incubation period of the disease (*e.g.*, past 14 days for COVID-19 [14,13]).

Definition 3 (Users). *A set U denotes all n users. Each user $u \in U$ is associated with a trajectory $L_u = \{(l_1, t_1), \ldots, (l_{|L_u|}, t_{|L_u|})\}$, following Definition 2, indicating the locations the user u visited and the time of the visits.*

Each user needs to be checked against the patients to see whether the user is a contact of some patient or not. The definition of a patient is similar to the one of the user, and we define them separately for easier illustration in the contact tracing problem.

Definition 4 (Patients). *A set P denotes all m patients. Each patient $p \in P$ has a trajectory $L_p = \{(l_1, t_1), \ldots, (l_{|L_p|}, t_{|L_p|})\}$, following Definition 2, indicating the locations the patient p visited and the time of the visits. The set $L_P = L_{p_1} \cup L_{p_2} \cup \ldots \cup L_{p_m}$ is a union of trajectories of all patients $p_1, \ldots, p_m$.*

In our problem setting, we use L_P to denote the aggregated set of all trajectories of patients. It is an important notion because a user $u \in U$ is defined as a *contact* (see a more formal definition in Definition 5) if the user u's trajectory L_u overlaps with some location in L_P. It is not necessary to identify which specific patient $p \in P$ leads to the contact. Indeed, as we will explain further in our adversary model and system settings in Sec. 2.2, it suffices to store the union of all trajectories L_P of the patients, without distinguishing each patient's individual trajectories, and it also enhances privacy protection.

Definition 5 (Contacts). *Given a distance threshold r, and a time difference threshold δ, a user $u \in U$ is called the* ***contact*** *if the user u has visited a location l_u that is within r distance to some visited location l_p of some patient $p \in P$, and the time difference between the two visits is within δ, i.e.,*

$$\exists (l_u, t_u) \in L_u \; \exists (l_p, t_p) \in L_P \; ((d_s(l_u, l_p) \leq r) \wedge (d_t(t_p, t_u) \leq \delta)) \tag{1}$$

where the function $d_s(l_u, l_p)$ represents the spatial distance – the Euclidean distance between locations l_u and l_p. $d_t(t_u, t_p)$ represents the temporal difference – the time difference in seconds from an earlier timestamp t_p to a later timestamp t_u.

We give a toy example in Example 1 to better illustrate the concept of contacts.

Example 1. As shown in Fig. 1b, there is one patient p_1 and two users u_1-u_2. User u_1 has a trajectory $L_{u_1} = \{(l_1, t_1), (l_2, t_2)\}$. User u_2 has a trajectory $L_{u_2} = \{(l_3, t_3), (l_4, t_4)\}$. The set of patients P (only contains one single patient p) has the trajectory $L_P = \{(l_5, t_5), (l_6, t_6)\}$. The locations are shown on the figure, and let us further specify the time. Let $t_1 =$ 2021-06-10 11:00:00 and $t_5 =$ 2021-06-10 10:00:00, indicating that patient p visits the location l_5 an hour earlier than the time when user u visits l_1.

Based on Definition 5, let r be the distance shown in the figure, and the time difference threshold be $\delta = 2$ hours, then the user u_1 is a contact, since $d_s(l_1, l_5) \leq r$, *i.e.*, u_1 has visited l_1 and l_1 is within distance r to patient p_1's visited location l_5, and $d_t(t_1, t_5) = 1$ hour $\leq \delta$, *i.e.*, the time difference between the two visits is within δ ($\delta = 2$ hours).

The temporal information plays a key role in the definition. If user u_1 visits the same location l_1 at a different time (*e.g.*, let $t_1 =$ 2021-06-10 13:00:00), then since $d_t(t_1, t_5) = 3$ hours $> \delta$, which violates the temporal constraint, u_1 would not be defined as a contact in that case.

On the other hand, user u_2 is not a close-contact, because her visited locations, l_3 and l_4, are not in proximity of any patient's visited locations.

2.2 Adversary Model

There are two major roles in our application: the users (the clients) and the government (the server). All trajectories by the patients (represented as L_P as in Definition 4) are aggregated and stored at the server. The setting follows the real-world situation: governments often collect whereabouts of confirmed patients in order to minimize any further transmission, usually starting with tracing the close contacts of the patients. For example, this is enforced by legislation in Hong Kong[2].

We adopt a semi-honest model as the adversary model in our problem. Adversary could exist on both the client (the user) and the server side (the government). We assume both the client and the server are curious about other

[2] https://www.elegislation.gov.hk/hk/cap599D!en?INDEX_CS=N

parties' private information, but they are not malicious and follow our designed system protocols. The semi-honest model is a commonly adopted setting in recent privacy-preserving LBS related applications [25,32,24].

2.3 Privacy-Preserving Contact Tracing Problem

Based on the concepts and adversary model introduced previously, we now define the Privacy-Preserving Contact Tracing (PPCT) problem as follows.

Definition 6 (Privacy-Preserving Contact Tracing (PPCT) problem). *Given a set U of n users , the trajectory L_u for each user $u \in U$, a set P of m patients, a set L_P containing the union of trajectories for all patients $p_1, \ldots, p_m \in P$, a distance threshold r, and a time difference threshold δ, the PPCT problem needs to correctly identify each user as a contact (see Definition 5) or not.*

In addition, PPCT has the following privacy requirements:

- *The computation process needs to be differentially private / confidential* w.r.t. *each user's trajectory L_u.*
- *The computation process needs to be differentially private / confidential* w.r.t. *L_P, the union of trajectories of the patients.*

With different trust assumptions and tradeoffs, there are a couple of choices for techniques which provide commonly accepted privacy guarantee: secure computation [30,27,4], trusted execution environments [3,33], and differential privacy [20,17].

3 Methodology

In this section, we first introduce two baselines, one based on Secure Multiparty Computation (MPC) and the other one based on Geo-Indistinguishability (Geo-I). Then, we introduce our proposed method – ContactGuard.

3.1 Baselines

MPC and Geo-I are two different paradigms to satisfy the privacy requirements in the PPCT problem, but they differ greatly in terms of efficiency and accuracy. MPC allows the server and the client to use cryptographic primitives to securely compare each visited location without revealing the exact locations of one party to the other parties. However, it induces significant computation overhead.

Geo-I [2] is an extended notion of differential privacy into the spatial domain (strictly speaking, Geo-I is a variant of local differential privacy notion [11,7]). It is an efficient mechanism to be applied to the location inputs, however Geo-I injects noise into the inputs (protecting the input location by perturbing it to an obfuscated location). Using the perturbed locations to compare trajectories would then lead to errors and reduce the accuracy of contact tracing.

MPC baseline The MPC baseline is to directly apply existing secure multi-party computation techniques to our problem. In our setting, each time, we treat a client (a user) and the server (which holds the patients' data) as two parties. Each of the party holds their own visited locations, which are private. Then, they use MPC operations to compare their visited locations and check whether the user is a contact or not, according to Definition 5. Note that the baseline computes the *exact* result, which means that it does not lose any accuracy.

The technique is enabled and implemented with Obliv-C [31], which is a language framework that provides high-level oblivious data structures for us to store and compare visited locations of users and the patients in an oblivious way. The underlying security protocol is provided by the language framework.

Geo-I baseline In Geo-I baseline, each user perturbs his/her trajectory L_u to a protected noisy location set L'_u using Geo-I (Algo. 1). Then, the user submits L'_u to the server. The server directly compares L'_u with the patients' locations L_P. If there exists some perturbed location $l' \in L'_u$ that is within distance r' (a system parameter), then u is identified as a contact.

Algorithm 1: `Perturb_Location_Set`

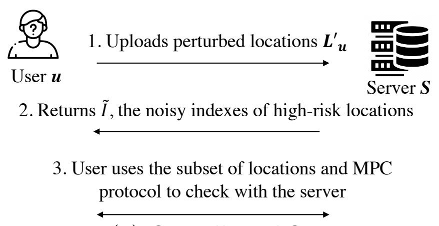
```
Input: $\epsilon, L_u$.
Output: $L'_u$.
1 $L'_u := \{\}$
2 $\epsilon' = \epsilon/|L_u|$
3 foreach $l \in L_u$ do
4     $l' = \text{Geo-I}(\epsilon', l)$
5     $L'_u$.insert($l'$)
6 return $L'_u$
```

Algo. 1 is a generic method to obtain a set of perturbed locations L'_u, given a set of original locations L_u and a total privacy budget of ϵ. At Line 2, the total privacy budget ϵ is equally divided into $|L_u|$ shares. Thus, each share is $\epsilon' = \epsilon/|L_u|$. Then we perturb each location $l \in L_u$ into a new location l' by the Geo-I mechanism. All perturbed locations l' compose the set L'_u of the perturbed locations.

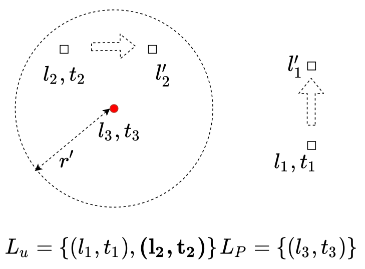
Note that in the original definition for the trajectory L_u in Definition 2, each visited location l is associated with a timestamp that the location is visited (see Example 1). We omit the timestamp information for each visited location, and perturb *only* the locations to perturbed locations. We abuse the notation L_u and L'_u to denote only the spatial locations (Definition 1) and the generated perturbed locations.

Time Complexity . For Algo. 1 (the client side), the time complexity is $\mathcal{O}(|L_u|)$, linear to the number of visited locations given in the input L_u. For the server side, because it compares each location of L'_u against every location of L_P, the time complexity is $\mathcal{O}(|L'_u||L_P|) \rightarrow \mathcal{O}(|L_u||L_P|)$. For the server S to finish the processing of all users, the total time complexity is thus $\mathcal{O}(\sum_u |L_u||L_P|) \rightarrow \mathcal{O}(|U||L_P| \max_u |L_u|)$.

Privacy Analysis. We defer the privacy analysis of Algo. 1 to Sec. 3.2, as it is the same as the privacy analysis of ContactGuard.


User u
1. Uploads perturbed locations L'_u
Server S
2. Returns $\tilde{I}$, the noisy indexes of high-risk locations
3. User uses the subset of locations and MPC protocol to check with the server

(a) Overall workflow.


l_2, t_2
l'_2
l_3, t_3
r'
l'_1
l_1, t_1
$L_u = \{(l_1, t_1), (\mathbf{l_2}, \mathbf{t_2})\}$ $L_P = \{(l_3, t_3)\}$

(b) Subset selection.

Fig. 2: The overall workflow and the subset selection process in Step 3.

3.2 Our Solution ContactGuard

In this section, we propose a novel method called ContactGuard. As we have previously introduced, the MPC baseline provides excellent accuracy, because it invokes computationally heavy MPC operations to compare all visited locations of users against the visited locations of the patients. The weakness of the MPC baseline is its poor efficiency. On the other hand, the Geo-I baseline is efficient because each client (user) applies an efficient Geo-I mechanism to change his/her visited locations to perturbed ones. But the inaccurate perturbed locations could lead to false positives/negatives for the PPCT problem.

The ContactGuard aims to combine the advantages of both baselines. On one hand, it uses the exact locations and MPC to determine whether a user is a contact or not to ensure excellent accuracy while providing strong privacy protection. On the other hand, it accelerates the MPC operation by selecting only a subset of users' visited locations to invoke the heavy MPC operations. The selection is done with the help of Geo-I. The overall workflow of ContactGuard is illustrated in Fig. 2a. The details are explained next.

Step 1. Location perturbation At this step, the user perturbs his/her visited location set L_u to a perturbed visited location set L'_u, and submits L'_u to the server. The steps are similar to the Geo-I baseline (Sec. 3.1).

Given a privacy budget ϵ for each user, and the true location set L_u, the user generates a perturbed location set L'_u using Algo. 1. Similar to the Geo-I baseline, the total privacy budget ϵ is equally divided into $|L_u|$ shares. Each location $l \in L_u$ is then perturbed to a noisy location l' by Geo-I mechanism with a privacy budget of $\epsilon' = \epsilon/|L_u|$.

Similar to the Geo-I baseline, we abuse notations L_u and L'_u to denote only the locations (without the timestamps) and the perturbed locations. This is slightly different from the original definition for the visited location set L_u in Definition 2, where each location is associated with a timestamp. We simply ignore all the temporal information during the perturbation process, as a way to provide strong privacy guarantee for the timestamps (as we do not use it at all). The temporal information will be used to check for the temporal constraints for determining a contact in later MPC steps (Step 3).

Privacy analysis. To understand the level of privacy guarantee of Algo. 1, we provide the theoretical privacy analysis for it. It extends Geo-I [2] from protecting a single location to a set of locations. The brief idea of the extension was mentioned in [2], we provide a concrete and formal analysis here.

First, we extend the privacy definition of Geo-I from a single location to a set of locations.

Definition 7. *(General-Geo-I) For two tuples of locations $\mathbf{l} = (l_1, \dots, l_n), \mathbf{l}' = (l'_1, \dots, l'_n)$, a privacy parameter ϵ, a mechanism M satisfies ϵ-General-Geo-I iff:*

$$d_\rho(M(\mathbf{l}), M(\mathbf{l}')) \leq \epsilon d_\infty(\mathbf{l}, \mathbf{l}'),$$

where $d_\infty(\mathbf{l}, \mathbf{l}') = max_i d(l_i, l'_i)$, which is the largest Euclidean distance among all pairs of locations from $\mathbf{l}$ and $\mathbf{l}'$.

Different from the Geo-I definition for a single location, here the distance between two tuples of locations is defined as the largest distance at any dimensions. If a randomized mechanism M satisfies the General-Geo-I in Definition 7, then given the any perturbed location set output $\mathbf{o} = (o_1, \dots, o_n)$, which is a sequence of perturbed locations generated by M, no adversary could distinguish whether the perturbed locations are generated from true location set $\mathbf{l}$ or from $\mathbf{l}'$. The ratio of the probabilities of producing the same output $\mathbf{o}$ from the true location set $\mathbf{l}$ or its neighboring input $\mathbf{l}'$ is bounded by $\epsilon d_\infty(\mathbf{l}, \mathbf{l}')$.

As a special case, when there is only one location in $\mathbf{l}$, Definition 7 becomes the original definition in Geo-I. We start the privacy analysis by showing the composition theorem for the General-Geo-I.

Lemma 1. *Let K_0 be a mechanism satisfying ϵ_1-Geo-I and K_1 be another mechanism satisfying ϵ_2-Geo-I. For a given 2-location tuple $\mathbf{l} = (l_1, l_2)$, the combination of K_0 and K_1 defined as $K_{1,2}(\mathbf{l}) = (K_1(l_1), K_2(l_2))$ satisfies $(\epsilon_1 + \epsilon_2)$-General-Geo-I (Definition 7).*

Proof. Let a potential output of the combined mechanism, $K_{1,2}(\mathbf{l})$, be $\mathbf{o} = (o_1, o_2)$, indicating a tuple of two perturbed locations. The probability of generating $\mathbf{o}$ from two input location sets $\mathbf{l} = (l_1, l_2)$ and $\mathbf{l}' = (l'_1, l'_2)$ are $\Pr[K_{1,2}(\mathbf{l}) = \mathbf{o}]$ and $\Pr[K_{1,2}(\mathbf{l}') = \mathbf{o}]$, respectively. Then, we measure the ratio of the two probabilities:

$$\begin{aligned}
\frac{\Pr[K_{1,2}(\mathbf{l}) = \mathbf{o}]}{\Pr[K_{1,2}(\mathbf{l}') = \mathbf{o}]} &= \frac{\Pr[K_1(l_1) = o_1] \cdot \Pr[K_2(l_2) = o_2]}{\Pr[K_1(l'_1) = o_1] \cdot \Pr[K_2(l'_2) = o_2]} && (2)\\
&= \frac{\Pr[K_1(l_1) = o_1]}{\Pr[K_1(l'_1) = o_1]} \cdot \frac{\Pr[K_2(l_2) = o_2]}{\cdot \Pr[K_2(l'_2) = o_2]} && (3)\\
&\leq e^{\epsilon_1 \cdot d(l_1, l'_1)} \cdot e^{\epsilon_2 \cdot d(l_2, l'_2)} && (4)\\
&\leq e^{\epsilon_1 \cdot d_\infty(\mathbf{l}, \mathbf{l}')} \cdot e^{\epsilon_2 \cdot d_\infty(\mathbf{l}, \mathbf{l}')} && (5)\\
&= e^{(\epsilon_1 + \epsilon_2) d_\infty(\mathbf{l}, \mathbf{l}')} && (6)
\end{aligned}$$

The end result in Equation (6) shows that $K_{1,2}$ satisfies $(\epsilon_1 + \epsilon_2)$-General-Geo-I (Definition 7).

Theorem 1. *Algo. 1 satisfies ϵ-General-Geo-I (Definition 7).*

Proof. We use the composition theorem of Geo-I in Lemma 1 to show that Algo. 1 composes linearly and consumes a total privacy budget of ϵ over the entire set of locations.

At Line 4 of Algo. 1, for each location $l \in L_u$, it is perturbed to l' with a privacy budget $\epsilon' = \epsilon_u/|L_u|$. Then, for each location, we achieve ϵ'-Geo-I. If there are two locations in L_u, and because each location is perturbed independently, we achieve $\epsilon' + \epsilon' = 2\epsilon'$-General-Geo-I. Then, by induction, for $|L_u|$ locations, we achieve $\sum_{l \in L_u} \epsilon' = \epsilon' \cdot |L_u| = \frac{\epsilon}{|L_u|} \cdot |L_u| = \epsilon$-General-Geo-I.

Step 2. Subset selection After Step 1, each user has a set of perturbed locations L'_u and submits it to the server. At Step 2, subset selection, upon receiving L'_u, the server selects a subset of high-risk locations based on the perturbed locations. This step is done at the server side, where the patients' true locations are stored. The server returns to the client the index of the selected high-risk locations (*e.g.*, if the 2nd location and the 4th location are high-risk locations, then the server returns $\{2, 4\}$) to indicate which location out of L_u are close to the patients. In order to protect the privacy of the patients, randomized response is used to perturb the indexes of high-risk locations to noisy indexes.

On the server side, the server does not know the true locations nor the timestamps of the visits, which are stored in L_u on the client side (the user). However, the server has access to the true locations visited by the patients, which are denoted as L_P. L_P are the basis to judge whether a user location is high-risk or not.

Theorem 2. *The subset selection step satisfies ϵ_P-Local Differential Privacy .*

Due to space limit, we defer the proof to the full version [1]. We also provide a detailed running example therein for this step.

Step 3. Accelerated MPC After Step 2, the user u receives the set $\tilde{I}$, which contains the indexes of high-risk locations as identified by the server. At Step 3, accelerated MPC, the user uses MPC protocol to check with the server about whether the user is a contact or not, using only the locations indicated by the set $\tilde{I}$. In contrast, the MPC baseline uses every location in L_u to check with the server using MPC protocol. Here, the user uses only a small subset of locations to perform secure computations.

Our method runs the same secure computation procedure as the MPC baseline. However, it only provides the subset of locations as the inputs to the `ProtocolIO` object `io`. The timestamps are also included in the *io* object, and are used to compare with the patients' trajectories L_P. This is different from the Geo-I baseline or Step 1. location perturbation in ContactGuard, which only uses the spatial information (the locations).

Overall Analysis Overall, for the server to finish processing all users, it takes $\mathcal{O}(\sum_u |I_u||L_P|) \to \mathcal{O}(|U||L_P|\max_u |I_u|)$, where I_u indicates the subset for each user u.
Privacy analysis. ContactGuard offers end-to-end privacy guarantee *w.r.t.* to both L_u and L_P. The detailed analysis is deferred to the full report [1].

Table 3 lists the comparison of the two baselines with our proposed ContactGuard. While achieving the same privacy and accuracy, our proposed solution significantly improves the efficiency of the MPC solution.

Table 1: Comparison of baselines and ContactGuard

Methods	**MPC baseline**	**Geo-I baseline**	**ContactGuard**
Accuracy (Recall)	100% $\surd$	39.0-70% $\times$	86-100% $\surd$
Efficiency	# of Secure Operations: $\mathcal{O}(\|U\|\|L_P\|\max_u \|L_u\|)$ $\times$	#of Plaintext Operations: $\mathcal{O}(\|U\|\|L_P\|\max_u \|L_u\|)$ $\surd$	# of Secure Operations: $\mathcal{O}(\|U\|\|L_P\|\max_u \|I_u\|)$ $\surd$
Privacy	Confidentiality $\surd$	Local-d DP $\surd$	Local-d DP $\surd$

4 Experimental Study

In this section, we first introduce the experimental setup in Sec. 4.1. Then, we present the detailed experimental results in Sec. 4.2.

4.1 Experimental setup

Datasets. We use a real-world dataset Gowalla [6] and a randomly generated synthetic dataset in our experiments.
Baselines. We compare our proposed ContactGuard (short as **CG**) method with the MPC baseline (Sec. 3.1, short as **MPC**) and the Geo-I baseline (Sec. 3.1, short as **Geo-I**).
Metrics. We focus on the following metrics:

- Recall: the number of found true close-contacts divided by the total number of close-contacts.
- Precision: the number of found true close-contacts divided by the number of predicted close-contacts.
- F_1 score: the harmonic mean of the precision and recall. $F_1 = 2\cdot\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}}$.
- Accuracy: the percentage of correctly classified users.
- Running time: the running time of the method in seconds.

Recall is considered as the most important metric in our experiments, as it measures the capability of the tested method for identifying potential close-contacts out of the true close-contacts, which is crucial in the application scenario of contact tracing of infectious disease.

Control variables.

We vary the following control variables in our experiments to test our method. Default parameters are in boldface. The privacy budget setting adopts the common setting with the real-world deployments (Apple and Google) [10].

- Number of users: $|U| \in [\mathbf{200}, 400, 800, 1600]$.
  Scalability test: $|U| \in [10K, 100K, 1M]$.
- Privacy budget for each user: $\epsilon \in [2.0, 3.0, \mathbf{4.0}, 5.0]$.
- Privacy budget for the patients: $\epsilon_P \in [2.0, 3.0, \mathbf{4.0}, 5.0]$.

4.2 Experimental Results

Our results show that the proposed solution ContactGuard achieves the best effectiveness/efficiency tradeoff, reducing the running time of the MPC baseline by up to 2.85×. The efficient Geo-I baseline, however, only provides poor effectiveness. Meanwhile, ContactGuard maintains the same level of effectiveness (measured by recall, precision, F_1 score and accuracy) as the one of the MPC baseline. The scalability tests also show that ContactGuard accelerates the MPC baseline significantly. When the number of users is large (*e.g.*, 500K), our ContactGuard supports daily execution, which is desirable in real-world application, while the MPC baseline fails to terminate within reasonable time (*e.g.*, 24 hours).

Effectiveness vs. Efficiency tradeoff.

Fig. 3 demonstrates the effectiveness and efficiency tradeoff of different methods. The y-axis are the running time of different methods (MPC baseline, Geo-I baseline and our proposed ContactGuard), and the x-axis are the respective measures for effectiveness (recall, precision, F_1 and accuracy).

As we could see from Fig. 3, the MPC baseline is an *exact* method, which obtains 100% in all measures of effectiveness. However, the high accuracy comes with prohibitive running time. It runs for 99.35 seconds when the number of users is 200 ($|U| = 200$) and 194.99 seconds for 400 users ($|U| = 400$).

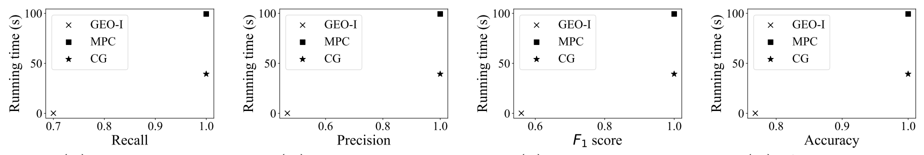
As a comparison, the Geo-I method is efficient, but comes with a significant decrease in effectiveness. For the setting $|U| = 200, \epsilon = 4.0$, Geo-I obtains about 70% recall and merely 46.7% precision. For the setting $|U| = 400, \epsilon = 3.0$, the recall drops to 39.0% and the precision drops to 42.0%.

Our proposed solution ContactGuard achieves the best effectiveness/efficiency tradeoff. For the setting $|U| = 200, \epsilon = 4.0$ (Fig. 3), CG obtains the same level of effectiveness as the one of the MPC baseline, achieving 100% in recall, precision and accuracy.

As compared to the Geo-I baseline, CG method obtains 2.28×, 2.38×, and 1.24× improvement in terms of recall, precision and accuracy, respectively. On the other hand, when we compare to the *exact* MPC baseline, the CG method is about 2.5× faster.

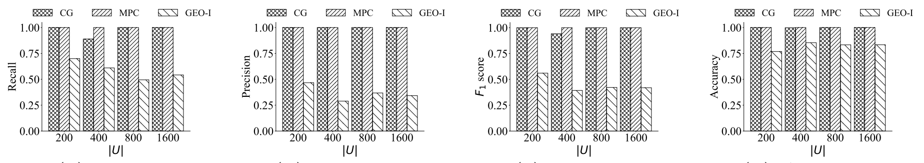
Effect of control variables.

Then, we test the effectiveness across different input sizes ($|U|$) and different values of the privacy budget (ϵ).



(a) Recall (b) Precision (c) F_1 score (d) Accuracy

Fig. 3: Effectiveness (Recall/Precision/F_1/Accuracy) vs. Running time trade-off of different methods on the Gowalla dataset. $|U| = 200, \epsilon = 4.0$.



(a) Recall. (b) Precision. (c) F_1 score. (d) Accuracy.

Fig. 4: Effectiveness (Recall/Precision/F_1/Accuracy) when varying the number of users ($|U|$) on the Gowalla dataset. $\epsilon = 4.0$.

Varying $|U|$: Fig. 4 shows measures of effectiveness (recall, precision, F_1, and accuracy) when we vary the number of users ($|U|$) for the Gowalla dataset. Across different input sizes, the CG method maintains the same level of recall/precision/F_1 score/accuracy as the *exact* MPC baseline across different $|U|$. This verifies that our CG method is highly effective across different input sizes.

On the other hand, the Geo-I baseline sacrifices significant effectiveness (in recall/precision/F_1/accuracy) across different $|U|$. The recall drops from 70% when $|U| = 200$ to 54.2% when $|U| = 1600$. The precision is constantly below 50%, and hovers around 35% when $|U|$ gets larger than 800. It shows that the pure differential privacy based solution injects noise to the perturbed locations, and the perturbed locations inevitably lead to poor effectiveness in the contact tracing applications.

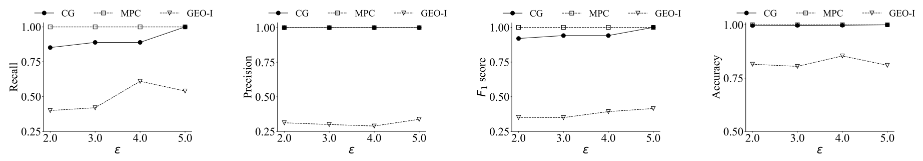
Varying ϵ: Fig. 5 demonstrates the impact of the privacy budget on the effectiveness (recall/precision/F_1/accuracy). In terms of recall, CG obtains the same level of effectiveness (reaching 100%) as the MPC baseline when $\epsilon = 5.0$. The recall is 85.2% when the privacy budget is small ($\epsilon = 2.0$) and improves to 88.9% when $\epsilon = 3.0$. In these two settings, the improvement over the Geo-I baseline are about 2.12×.

In terms of precision, CG never introduces false positives, because positive results are returned only when there is indeed a true visited location of the user (included in the subset selected and verified with MPC operation) which overlaps with the patients. Thus, the precision is constantly at 100% over different ϵ.

The F_1 score reflects the similar pattern as the recall, as it is an average of recall and precision. The accuracy of CG stays at more than 99% across different ϵ.

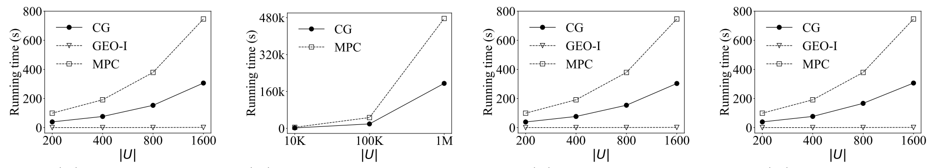
Efficiency and scalability.

When compared to the MPC baseline, our solution CG achieves significant speed up. Fig. 6 shows the running time of different methods across different input sizes, when the number of users scales up to 1 million.



(a) Recall. (b) Precision. (c) F_1 score. (d) Accuracy.

Fig. 5: Effectiveness (Recall/Precision/F_1/Accuracy) when varying the privacy budget (ϵ) on the Gowalla dataset. $|U| = 400$.



(a) $\epsilon = 4.0$. (b) Scalability tests. (c) $\epsilon = 2.0$. (d) $\epsilon = 3.0$.

Fig. 6: Efficiency (running time in seconds) when varying the number of users ($|U|$) on the synthetic dataset.

When the number of users ($|U|$) increases from 200 to 1600, the running time of the MPC baseline grows from 99.35 seconds to 746.34 seconds. The running time grows linearly with $|U|$.

In comparison, the running time of CG grows from 39.31 seconds to 306.20 seconds, also with a linear growth with $|U|$. The speed up of CG over MPC is 2.52×, 2.50×, 2.48×, and 2.44× when $|U|$ increases from 200 to 1600. The results are shown in Fig. 6a.

Then, we use the synthetic dataset to compare the scalability of the MPC baseline and the CG method. The results are shown in Fig. 6b. The running time for MPC is 4754.7 seconds, 46,913 seconds (13 hours) and 476,504.6 seconds (5.5 days) for $|U|$ =10K, 100K, and 1M. As a comparison, the CG method runs in 1936.8 seconds, 19,087.2 seconds (5.3 hours) and 195,060.7 seconds (2.25 days). It maintains a 2.5× speed up over the MPC baseline. This verifies that the subset selection in CG significantly reduces the running time of the MPC protocol. When the number of users reaches 1 million, the MPC baseline takes about 5.5 days to process all the users, while our proposed CG finishes in about 2 days.

5 Related Work

In this section, we review the related work based on the adopted privacy-preserving techniques.

Our work is closely related to REACT [9], as it also uses Geo-I [2] to preserve the location privacy. In REACT, each location is applied with Geo-I with the same privacy budget. As shown in the experiments (both in [9] and in our results), the Geo-I baseline performs poorly in terms of both recall and precision, because it only uses perturbed locations. As a similar notion to Geo-I, local differential privacy has also been used to protect the trajectory data [8]. However, its main use is for aggregated statistics over relatively larger regions, where in our application, each user's contact tracing and fine-grained visited location is important.

Many other privacy-preserving techniques are used for contact-tracing, including blockchain based P^2B-Trace [21] and cryptographic primitives enabled BeeTrace [19] and Epione [26]. As opposed to using relative locations in these works, our setting uses absolute locations, which enables tracing indirect contacts (transmission via a contaminated environment) in addition to direct contacts.

On topics not related to contact tracing, there are many recent works on combining differential privacy and cryptographic techniques [28]. Secure shuffling and encryption-based techniques [12,23] could help reduce the error introduced by differential privacy, and differential privacy could enable more efficient secure computation [5]. Our solution, ContactGuard, is inspired by this line of ideas, and uses Geo-I to accelerate MPC computations.

6 Conclusion

In this work, we study the Privacy-Preserving Contact Tracing (PPCT) problem, to identify the close-contacts of the patients while preserving the location privacy of the patients and the users. To address this problem, we propose an accurate and efficient solution named ContactGuard, which accelerates the Secure Multiparty Computation (MPC) with the help of Geo-Indistinguishability (Geo-I). Experimental results demonstrate that ContactGuard provides significant speed up over the MPC baseline, while maintains an excellent level of effectiveness (as measured by recall and precision).

References

1. Anonymous full paper. https://anonymous.4open.science/r/dpcovid_code-FF71/full.pdf
2. Andres, M.E., Bordenabe, N.E., Cjhatzikokolakis, K., Palamidessi, C.: Geo-indistinguishability: differential privacy for location-based systems. In: CCS (2013)
3. Arasu, A., Kaushik, R.: Oblivious query processing. ICDT (2014)
4. Bater, J., Elliott, G., Eggen, C., Goel, S., Kho, A.N., Rogers, J.: SMCQL: Secure query processing for private data networks. VLDB (2017)
5. Bater, J., He, X., Ehrich, W., Machanavajjhala, A., Rogers, J.: Shrinkwrap: Efficient SQL query processing in differentially private data federations. VLDB (2018)
6. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. KDD (2011)
7. Cormode, G., Kulkarni, T., Srivastava, D.: Answering range queries under local differential privacy. VLDB (2019)
8. Cunningham, T., Cormode, G., Ferhatosmanoglu, H., Srivastava, D.: Real-world trajectory sharing with local differential privacy. VLDB (2021)
9. Da, Y., Ahuja, R., Xiong, L., Shahabi, C.: REACT: real-time contact tracing and risk monitoring via privacy-enhanced mobile tracking. ICDE (2021)
10. Domingo-Ferrer, J., Sánchez, D., Blanco-Justicia, A.: The limits of differential privacy (and its misuse in data release and machine learning). Communications of the ACM **64**(7), 33–35 (2021)

11. Duchi, J.C., Jordan, M.I., Wainwright, M.J.: Local privacy and statistical minimax rates. In: FOCS (2013)
12. Erlingsson, Ú., Feldman, V., Mironov, I., Raghunathan, A., Talwar, K., Thakurta, A.: Amplification by shuffling: From local to central differential privacy via anonymity. SODA (2019)
13. Ferretti, L., Wymant, C., Kendall, M., Zhao, L., Nurtay, A., Abeler-Dörner, L., Parker, M., Bonsall, D., Fraser, C.: Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. Science **368**(6491) (2020)
14. Guan, W.j., Ni, Z.y., Hu, Y., et al.: Clinical characteristics of coronavirus disease 2019 in china. New England Journal of Medicine **382**(18), 1708–1720 (2020)
15. Hope, C.: Intel 11th and 12th gen chips can't play 4k blu-rays. PCGamer (2022)
16. Kato, F., Cao, Y., Yoshikawa, M.: Secure and efficient trajectory-based contact tracing using trusted hardware. In: IEEE Big Data (2020)
17. Kotsogiannis, I., Tao, Y., He, X., Fanaeepour, M., Machanavajjhala, A., Hay, M., Miklau, G.: Privatesql: a differentially private sql query engine (2019)
18. Lewis, D.: Why many countries failed at covid contact-tracing-but some got it right. Nature pp. 384–387 (2020)
19. Liu, X., Trieu, N., Kornaropoulos, E.M., Song, D.: Beetrace: A unified platform for secure contact tracing that breaks data silos. IEEE Data Eng. Bull. (2020)
20. McSherry, F.D.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. SIGMOD (2009)
21. Peng, Z., Xu, C., Wang, H., Huang, J., Xu, J., Chu, X.: P^2b-trace: Privacy-preserving blockchain-based contact tracing to combat pandemics. SIGMOD (2021)
22. Rodríguez, P., Graña, S., Alvarez-León, E.E., Battaglini, M., Darias, F.J., Hernán, M.A., López, R., Llaneza, P., Martín, M.C., Ramirez-Rubio, O., et al.: A population-based controlled experiment assessing the epidemiological impact of digital contact tracing. Nature communications **12**(1), 1–6 (2021)
23. Roy Chowdhury, A., Wang, C., He, X., Machanavajjhala, A., Jha, S.: Crypte: Crypto-assisted differential privacy on untrusted servers. SIGMOD (2020)
24. Tao, Q., Tong, Y., Zhou, Z., Shi, Y., Chen, L., Xu, K.: Differentially private online task assignment in spatial crowdsourcing: A tree-based approach. ICDE (2020)
25. To, H., Shahabi, C., Xiong, L.: Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. ICDE (2018)
26. Trieu, N., Shehata, K., Saxena, P., Shokri, R., Song, D.: Epione: Lightweight contact tracing with strong privacy. IEEE Data Eng. Bull. (2020)
27. Volgushev, N., Schwarzkopf, M., Getchell, B., Varia, M., Lapets, A., Bestavros, A.: Conclave: secure multi-party computation on big data. In: EuroSys (2019)
28. Wagh, S., He, X., Machanavajjhala, A., Mittal, P.: Dp-cryptography: marrying differential privacy and cryptography in emerging applications. Communications of the ACM **64**(2), 84–93 (2021)
29. Woo, R., Tham, E.: Chinese city says it mass tested 30,000 for covid-19 at mega centre, rounded-up runaways. Reuters (2021)
30. Yao, A.C.C.: How to generate and exchange secrets. In: FOCS (1986)
31. Zahur, S., Evans, D.: Obliv-c: A language for extensible data-oblivious computation. IACR Cryptol. ePrint Arch. **2015**, 1153 (2015)
32. Zhai, D., Sun, Y., Liu, A., Li, Z., Liu, G., Zhao, L., Zheng, K.: Towards secure and truthful task assignment in spatial crowdsourcing. WWW (2019)
33. Zheng, W., Dave, A., Beekman, J.G., Popa, R.A., Gonzalez, J.E., Stoica, I.: Opaque: An oblivious and encrypted distributed analytics platform. NSDI (2017)

Algorithm 2: Geo-I[2]

```
Input: $\epsilon, l = (x, y)$.
Output: $l'$.
1 draw $\theta$ uniformly in $[0, 2\pi)$
2 draw $p$ uniformly in $[0, 1)$
3 $d = -\frac{1}{\epsilon}(W_{-1}(\frac{p-1}{e}) + 1)$
4 $x' = l.x + \cos(\theta) \cdot d$
5 $y' = l.y + \sin(\theta) \cdot d$
6 return $l' = (x', y')$
```

A Additional technical details

For easy reference, the major notations used in this paper are summarized in Table 2.

Table 2: Major notations used in this paper.

Symbol	Description
u, U	A user and the set of n users
p, P	A patient and the set of m patients
l, l'	A location and its perturbed location
L_u, L'_u	The trajectories of user u and its perturbed locations
L_P	The set of trajectories for all the patients
$I, \tilde{I}$	The index of high risk locations and the noisy indexes
r	Distance threshold for determining a close contact
r'	Our system parameter for determining a high-risk location
ϵ, ϵ_P	The privacy budget for the user and the patients

A.1 Application Settings and Adversary Model

Application settings We model the contact tracing application as a client-server model (illustrated briefly in Fig. 1a):
Client side. Each user $u \in U$ corresponds to a client. Each user stores his/her own locations L_u locally on a mobile device.
Server side. All trajectories by the patients (represented as L_P as in Definition 4) are aggregated and stored at the server. In our contact tracing application, the server is owned by the government, or more specifically, a central public health organization (*e.g.*, the Centers of Disease Control and Prevention). The setting follows the real-world situation: governments often collect whereabouts of confirmed patients in order to minimize any further transmission, usually starting with tracing the close contacts of the patients. For example, this is enforced by legislation in Hong Kong[3].

[3] https://www.elegislation.gov.hk/hk/cap599D!en?INDEX_CS=N

For the purpose of contact tracing, rather than storing each patient's individual trajectories, it suffices to store the union of trajectories of all patients (represented as L_P). This is because given a user, our goal is only to determine whether the user co-visits the nearby location as *some patient* $p \in P$, without the need to identify specifically which patient. This partially enhances privacy protection, as no individual patient's trajectory is identified and stored.

As a summary, there are two major roles in our application: the users (the clients) and the government (the server).

For the adversary on the server, the adversary tries his/her best to obtain as much private information as possible from the users (obtaining information about the locations L_{u_i}). Under this model, our proposed solution needs to limit the information shared from the user to the server. When the information about L_u is needed, the computation process needs to be protected with privacy guarantee, *w.r.t.* L_u.

Similarly, adversary could exist on the client side. It means that the adversary tries his/her best to obtain as much private information as possible from the server (which stores the patients' locations L_P) or from other users (obtaining information about other users' L_u). Thus, our proposed solution needs to limit the information shared from the server to each user as well. When the information about L_P is needed, the computation process needs to be protected with privacy guarantee, *w.r.t.* L_P.

Performance metrics Considering our specific application – contact tracing, we measure the quality of a solution based on the following criteria:
(1) Effectiveness . We use recall, precision and accuracy as the key metrics for measuring the effectiveness (utility) of a potential solution. In the contact tracing applications, *recall* is defined as the ratio of the correctly identified contacts among all contacts. *Precision* is defined as the ratio of correctly identified contacts among all contacts identified by the solution. The *accuracy* is defined as the ratio of correctly classified instances (both the correctly identified contact and the non-contacts) over all users.

While all three metrics are crucial, recall is considered relatively more important in the contact tracing application, as it measures the capability of a solution for eliminating false negatives (identifying as many true contacts as possible). This is crucial in applications of epidemic control, where missing a true contacts may lead to further uncontrolled transmission of the disease. Taking the epidemic control of COVID-19 in China as an example, when several cases were found in a district, all citizens of the entire city need to take mandated testing in order to minimize the chances of missing any positive cases [29].
(2) Efficiency . We use the end-to-end running time as the metric to measure the efficiency of a proposed solution. The end-to-end running time is the time needed to process all users in U to determine whether they are contacts or not. As the contact tracing application is normally executed on a daily basis, if a solution could not process all users in 24 hours, it is considered impractical in terms of efficiency.

Algorithm 3: `MPC_Compare_Location`

```
Input: $r, \delta$, ProtocolIO $io$.
Output: True/False.
1  $n_1$ := ocBroadcast($io$.n, party=1)
2  $n_2$ := ocBroadcast($io$.n, party=2)
3  obliv float ox_array1 = ocToOblivFloatArray(io.x, party=1)
4  obliv float ox_array2 = ocToOblivFloatArray(io.x, party=2)
5  obliv float oy_array1 = ocToOblivFloatArray(io.y, party=1)
6  obliv float oy_array2 = ocToOblivFloatArray(io.y, party=2)
7  obliv int time_array1 = ocToOblivIntArray(io.time, party=1)
8  obliv int time_array2 = ocToOblivIntArray(io.time, party=2)
9  foreach $i \in [0 \dots n_1 - 1]$ do
10     foreach $j \in [0 \dots n_2 - 1]$ do
11         obliv float d_x := ox_array1[i] - ox_array2[j]
12         obliv float d_y := oy_array1[i] - oy_array2[j]
13         obliv float d_square := d_x * d_x + d_y*d_y
14         obliv int $t_1$ :=time_array1[i]
15         obliv int $t_2$ :=time_array2[j]
16         obliv int diff_time := time_difference($t_1, t_2$)
17         obliv if (d_square <= $r * r$ AND diff_time <= $\delta$) then
18             return True
19 return False
```

A.2 Baselines

Table 3 lists the comparison of the two baselines.

Table 3: Comparison of baselines and ContactGuard

Methods	MPC	Geo-I	ContactGuard
Accuracy	√	×	√
Efficiency	×	√	√

Algo. 3 gives the detailed steps for the MPC baseline. We only show the main procedure that is shared and called by both the server (the patients) and the client (the user). The inputs to the procedure includes an object *io* of type `ProtocolIO`. The server side's input object *io* contains the trajectories L_P for all the patients. The client side's input object *io* holds the trajectory L_u for user u. Algo. 3 uses secure computation over the private input *io* to determine whether the user u (the client side) is a contact or not.

From Line 3 to Line 8, the routine `ocToOblivFloatArray` in Obliv-C is called to transform the private inputs (an array with `float` type) to oblivious data types to be used during the secure computation. Similarly, the routine `ocToOblivIntArray` is to transform an integer arrays to an oblivious data type. In our implementation, we store the x and y coordinates of all visited locations in float arrays (`io.x` and `io.y`). We store the timestamps as integer arrays.

Note that when we call the routine `ocToOblivFloatArray`, an argument `party=1` is given to obtain the private inputs from the patients (the server, which we refer to as the 1st party). When `party=2` is given, the routine obtains the private inputs from the user (which we refer to as the 2nd party). After the private inputs are transformed to the oblivious data types, all computations based on these oblivious data types are guaranteed to be oblivious as well.

The number of visited locations $|L|$ is stored in `io.n` (an integer). As we do not consider the number of visited locations as a private input, it is broadcasted and stored as n_1 ($|L_P|$ for the patients) and n_2 ($|L_u|$ for the user).

Then we iterate over all location from the patients and all locations from the user to check whether the pair of visited locations and timestamps satisfy the spatial and temporal constraints in the contact definition in Definition 5. All computations are oblivious, including calculating the square of the Euclidean distance between the two locations and calculating the time difference in seconds between the two timestamps, because these computations are based on oblivious inputs (*e.g.*, ox_array1 which is transformed from the private inputs), and the operations only involve oblivious summations and multiplications, which are provided by the underlying Obliv-C.

At the last step (Line 17), the square of the Euclidean distance is compared with $r * r$, which is the square of the distance threshold r. In addition, the time difference in seconds is compared with δ, which is the time difference threshold. r and δ are inputs to our PPCT problem, as specified in Definition 6. The comparison uses the security primitive `obliv if` provided by Obliv-C, such that the execution of the program (the program counters or the running time) does not disclose any information about the underlying private inputs (the oblivious variables). The body of *obliv if* is executed regardless of the value of conditions.

Time Complexity. The oblivious transformation from private inputs to oblivious data types from Line 3 to Line 8 takes $\mathcal{O}(n_1 + n_2)$ to complete. The main overhead occurs when we iterate over all locations from the patients and all locations from the user, which takes $\mathcal{O}(n_1 * n_2) \rightarrow \mathcal{O}(|L_P||L_u|)$ to complete. Overall, for each user, the MPC procedure Algo. 3, takes $\mathcal{O}(|L_P| + |L_u| + |L_P||L_u|) \rightarrow \mathcal{O}(|L_P||L_u|)$. For the server S to finish processing all users, it takes $\mathcal{O}(\sum_u |L_u||L_P|) \rightarrow \mathcal{O}(|U||L_P| \max_u |L_u|)$.

Note that here the time complexity analysis assumes that the security primitives (*e.g.*, `ocToOblivFloat`, oblivious summation and multiplication, `oblivious if`) run in constant time ($\mathcal{O}(1)$). In practice, the security primitives run multiple orders of magnitude slower than running a non-secure statement. We stay at a high-level analysis in terms of the total number of execution of security primitives rather than evaluating the number of gates of the lower-level secure circuits that are generated by Obliv-C. In our proposed method ContactGuard, we optimize the running time by significantly reducing the number of secure primitives used.

Privacy Analysis.

Theorem 3. *Algo. 3 satisfies the privacy requirements in Definition 6.*

Proof. The privacy is obviously guaranteed since we are using MPC (provided by Obliv-C) operations over private inputs. We show that other than the size of the visited location set, Algo. 3 discloses nothing about L_u for the user and L_P for the patients.

About L_u: from Line 3 to Line 8, are transformed to oblivious data types. The computations inside the iterations from Line 9 to Line 18 only use the oblivious variables (*e.g.*, ox_array2). Thus, L_u is strictly protected. Note that in our problem setting (Definition 6), the length of the trajectory ($|L_u|$) is not considered sensitive, and in fact we broadcast $|L_u|$ at Line 2, and it is publicly available to both parties.

About L_P: from Line 3 to Line 8, are transformed to oblivious data types. The computations inside the iterations from Line 9 to Line 18 only use the oblivious variables (*e.g.*, ox_array1 and time_array1). Thus, L_P is strictly protected.

A.3 ContactGurad

It consists of the following steps.

Step 1. Location perturbation: each user perturbs his/her visited location set L_u to a perturbed visited location set L'_u. The user submits L'_u to the server.

Step 2. Subset selection: the server compares L'_u with L_P, the trajectories of the patients. The server keeps track of the indexes of the high-risk locations (*e.g.*, the 3rd/4th location) that are within distance r' to some visited location of the patients. Then, the server returns the list of indexes (denoted as $\tilde{I}$, with privacy protection) of the identified high-risk locations to the user.

Step 3. Accelerated MPC: the user selects the subset of visited locations based on $\tilde{I}$, which is returned by the server in Step 2, and then uses MPC protocol to compare the subset of locations with the patients' trajectories L_P on the server. If there is a location in the subset (together with the visited timestamp) that meets the definition of contact as Definition 5, the user u will be identified as a contact. Otherwise, the user will not be identified as a contact.

Proofs Equation (2) is by the definition of the combined mechanism $K_{1,2}(\mathbf{l}) = (K_1(l_1), K_2(l_2))$, which applies mechanism K_1 to the first location l_1 and K_2 to the second location l_2 independently. So, the probability of generating o_1 and o_2 are the products of probability of generating o_1 by K_1 and o_2 by K_2.

Equation (4) is because K_1 satisfies ϵ_1-Geo-I (Definition **??**) and K_2 satisfies ϵ_2-Geo-I. By the definition of Geo-I in Definition **??**, the probability of generating the same output location, o_1, from two input locations l_1 and l'_1 are bounded by $e^{\epsilon_1 \cdot d(l_1, l'_1)}$. The multiplicative distance between two distributions, $d_\rho(M(x), M(x'))$ in Equation (**??**) is defined as $|\ln \frac{M(x)}{M(x')}|$ (see the original paper [2]), here we transform the ln function back to the exponential form.

Equation (5) is by the definition of $d_\infty(\mathbf{l},\ d_\infty(\mathbf{l}, \mathbf{l'}) = max_i d(l_i, l'_i)$. Both $d(l_1, l'_1) \le max_i d(l_i, l'_i)$ and $d(l_2, l'_2) \le max_i d(l_i, l'_i)$ hold.

Privacy analysis. The privacy guarantee of this step is provided with the following theorem.

Proof. Without loss of generality, let us focus on one specific location l in L_P. Assume that l is within distance r' to the i-th location in L'_u. Then the exact index constructed is $\mathbf{i} = \{0, \ldots, 1, \ldots, 0\}$, where the i-th location is with value 1 and all the other positions are 0. Then, with the randomized response mechanism, for a given output $\mathbf{o} = \{0, \ldots, 1, \ldots, 0\}$, the probability of generating this output at the i-th position is $e^{\epsilon_P}/(e^{\epsilon_P}+1)$ (ignoring the probability of other positions as they are the same for the neighboring inputs L'_P, and will be cancelled out).

For the neighboring inputs L'_P, where all positions are the same except for $l' \in L'_P$, and l' be outside distance r' of the i-th location in L'_u and all the other locations in L'_u. The exact index constructed for this neighboring input is $\mathbf{i}' = \{0, \ldots, 0, \ldots, 0\}$, where the i-th position is with value 0. Then the probability of generating the same output $\mathbf{0} = \{0, \ldots, 1, \ldots, 0\}$ at the i-th position is $1/(e^{\epsilon_P}+1)$ (again ignoring the probabilities for the other positions). So, the ratio of the two probabilities are:

$$\frac{\Pr[R(\mathbf{i}) = \mathbf{o} | l \in L_P)]}{\Pr[R(\mathbf{i}') = \mathbf{o} | l' \in L'_P)]} = \frac{e^{\epsilon_P}/(e^{\epsilon_P}+1)}{1/(e^{\epsilon_P}+1)} = e^{\epsilon_P}. \tag{7}$$

Thus, the subset selection step satisfies ϵ_P-Local Differential Privacy.

Algorithms We use a running example next to better illustrate the basic idea of subset selection.

Example 2. The example is shown in Fig. 2b. In this example, the user u has visited location l_1 at time t_1 and l_2 at time t_2. At Step 1. location perturbation, the user perturbs location l_1 to l'_1 and l_2 to l'_1. The set L'_u comprises these two perturbed locations and is sent to the server.

In this example, the patient visited location l_3 at time t_3. Then, the server compares L'_u, which contains l'_1 and l'_2, with l_3. Because l'_2 locates close enough to l_3 (within a distance r'), l'_2 is identified as a high-risk location (and chosen to be included in the subset), whereas l'_1 is not chosen. Then the server returns the indexes of the subset of high-risk locations, denoted by $I = \{2\}$ back to the user, indicating that the 2nd location is a high-risk location.

For privacy reason, before the server returns the *exact* high-risk indexes, it uses randomized response to perturb them. Then, the noisy indexes are returned to the user. In this example, $I = \{2\}$ could also be represented as $\mathbf{i} = \{0, 1\}$ indicating the 1st location is not a high-risk location (indicated by a value of 0) and the 2nd location is high-risk (indicated by a value of 1). Each bit is perturbed with randomized response. Given a privacy budget ϵ_P, the original value is returned with a probability $e^{\epsilon_P}/(e^{\epsilon_P}+1)$. Otherwise, it returns a flipped value (0 to 1, 1 to 0) with probability $1/(e^{\epsilon_P}+1)$.

Algo. 4 describes the detailed steps to select the subset of high-risk locations. It initializes the returned index set as the empty set, and then for each perturbed location l', if it is within distance of r' to some location l visited by the patients

Algorithm 4: `Subset_Selection`

```
Input: $L'_u, L_P, r', \epsilon_P$.
Output: $\tilde{I}$.
1  $\tilde{I} := \{\}$
2  $i := 0$
3  $\gamma := e^{\epsilon_P}/(e^{\epsilon_P}+1)$
4  foreach $l' \in L'_u$ do
5      $i = i + 1$
6      flag:= 0
7      foreach $l \in L_P$ do
8          if $d(l', l) \leq r'$ then
9              flag = 1
10     Sample $b :=$ Bernoulli($\gamma$)                // Randomized response
11     if $b$ = 1 then
12         flag = 1 - flag                          // Flip the result
13     if flag = 1 then
14         $\tilde{I}$.insert($i$)
15 return $\tilde{I}$
```

(stored in L_P). For each location, we use randomized response to transform the original value (variable `flag`) to a noisy one. In the end, the noisy index $\tilde{I}$ is returned to the user u.

Note that the parameter r' is a controllable variables to balance the accuracy and efficiency trade-off of ContactGuard. If r' is set as a large value, then all locations would be included in the subset of high-risk locations and later be used in the MPC computations, and thus it would lead to a high overhead. If r' is set as a small value, then the perturbed location needs to be close to some patients' visited location in order to be included in the subset of high-risk locations, which would then lead to more false negatives.

Algo. 5 describes the steps to prepare the private inputs using only the subset of the locations for the MPC protocol.

Time complexity We conduct time complexity analysis for each step of ContactGuard.

Step 1, location perturbation: the client side (the user) runs Algo. 1. The time complexity is $\mathcal{O}(|L_u|)$, linear to the number of visited locations given in the input L_u.

Step 2, subset selection: the server side conducts the subset selection. The time complexity is $\mathcal{O}(|L'_u||L_P|) \rightarrow \mathcal{O}(|L_u||L_P|)$, because it compares each location of L'_u against every location of L_P.

Step 3, accelerated MPC: let $|I|$ be the size of the subset selected in Step 2. Then, the oblivious transformation from private inputs to oblivious data types takes $\mathcal{O}(|n_I| + |L_P|)$ to complete. The main overhead occurs when we iterate over all locations from the patients and only locations from the subset of the

Algorithm 5: `Prepare_MPC_Inputs`

```
Input: $L_u, \tilde{I}$.
Output: ProtocolIO $io$.
1 $io$ := Initialize a ProtocolIO object
2 $io.n = \tilde{I}$.size()
3 $i := 0$
4 foreach $l_{idx} \in \tilde{I}$ do
5     $io.x[i] = L_u[l_{idx}].x$
6     $io.y[i] = L_u[l_{idx}].y$
7     $io.time[i] = L_u[l_{idx}].time$
8     $i = i + 1$
9 return $io$
```

user's visited locations, which takes $\mathcal{O}(|L_P||I|)$ to complete. Overall, for each user, the MPC procedure takes $\mathcal{O}(|L_P| + |I| + |L_P||I|) \to \mathcal{O}(|L_P||I|)$.

Overall, for the server to finish processing all users, it takes $\mathcal{O}(\sum_u |I_u||L_P|) \to \mathcal{O}(|U||L_P| \max_u |I_u|)$, where I_u indicates the subset for each user u. The time complexity takes one MPC operation as a unit operation, and it overrides the $\mathcal{O}(|L_u||L_P|)$ time needed at Step 2, subset selection, which does not require secure computation.

Privacy analysis We show the end-to-end privacy analysis for ContactGuard.

Step 1, location perturbation: L_u is protected with ϵ-General-Geo-I as shown in Theorem 1. The client side does not have knowledge about the patients' locations (L_P), so Step 1 is private *w.r.t.* both L_u and L_P.

Step 2, subset selection: the computation is only done on the perturbed location L'_u. According to the post-processing property of Geo-I, the computation process is still private *w.r.t.* L_u. About the patients' locations L_P, Theorem 2 shows that the returned noisy index satisfies ϵ_P-LDP *w.r.t.* L_P. Thus, this step is also private *w.r.t.* both L_u and L_P.

Step 3, accelerated MPC: the privacy is guaranteed by the MPC procedure (provided by Obliv-C) operations over private inputs. On the client side, the subset of L_u is selected from L_P with the noisy index $\tilde{I}$, and then they are transformed to the oblivious data types. On the server, L_P is also transformed to the oblivious data types. All further computations are based on oblivious operators.

B Additional experimental results

Gowalla dataset: it is a location-based social network check-in dataset [6], recording the latitude and the longitude of users' visited locations. The dataset contains records in different cities around the world, and we select the records from San Francisco city, allowing the GIS coordinates to be mapped to X,Y Cartesian coordinates in the range of $[0, 10549] \times [0, 8499]$, such that the Geo-I could be

Table 4: Recall/Precision/Accuracy vs. Running time trade-off of different methods on the Gowalla dataset. $|U| = 200, \epsilon = 4.0$.

Method	Geo-I	MPC	ContactGuard
Recall	70.0%	100%	**100%**
Precision	46.7%	100%	**100%**
F_1 score	56.0%	100%	**100%**
Accuracy	76.8%	100%	**100%**
Running time (s)	0.08	99.35	**39.31 (2.53x faster)**

Table 5: Recall/Precision/Accuracy vs. Running time trade-off of different methods on the Gowalla dataset. $|U| = 400, \epsilon = 3.0$.

Method	Geo-I	MPC	ContactGuard
Recall	39.0%	100%	**88.89%**
Precision	42.0%	100%	**100%**
F_1 score	35.0%	100%	**94.1%**
Accuracy	80.5%	100%	**99.75%**
Running time (s)	0.39	194.99	**77.03 (2.52x faster)**

directly applied to the X,Y coordinates. We then randomly select a small subset of the users as the patients (the number of patient is set as 2 to 8, with the patient/user ratio set as around 1%), and the rest are set as users to be tested.

Synthetic dataset: we generate each location with X,Y coordinates from the range $[0, 10549] \times [0, 8499]$, which is the same as the one of the Gowalla dataset.

System configuration The experiment is conducted on a CentOS Linux system with Intel(R) Xeon(R) Gold 6240R CPU @2.40GHz and 1007G memory. We implement the methods and conduct the experiments using C. The MPC extension is provided by Obliv-C [31]. We use two processes on the same machine to simulate the client-server MPC protocol, and neglect the communication time.

The detailed results are shown in Table 4 and Table 5, respectively.

As compared to the Geo-I baseline, CG method obtains 2.28×, 2.38×, and 1.24× improvement in terms of recall, precision and accuracy, respectively. On the other hand, when we compare to the *exact* MPC baseline, the CG method is about 2.5× faster (in both Table 4 and Table 5).

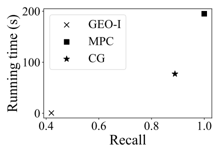
For a different setting where the privacy budget is smaller ($|U| = 400, \epsilon = 3.0$, as in Fig. 7), CG obtains a reasonably high level of effectiveness. In addition to maintaining a 100% precision, it obtains 88.89% in recall and 99.75% in accuracy.

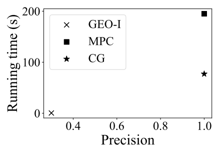
Fig. 8 shows measures of effectiveness (recall, precision, F_1, and accuracy) when we vary the number of users ($|U|$), for the synthetic dataset.

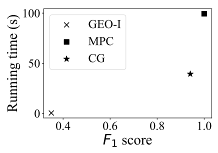
B.1 More results on the Gowalla dataset

More on varying ϵ_P:

The CG method obtains high effectiveness as measured by recall/precision/F_1/accuracy when we vary ϵ_P. Though the recall drops to around 0.8


(a) Recall vs. Running time.


(b) Precision vs. Running time.


(c) F_1 score vs. Running time.

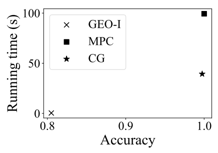
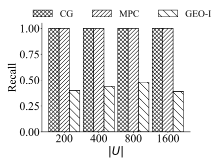
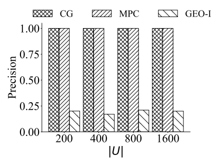
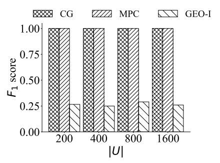

(d) Accuracy vs. Running time.

Fig. 7: Effectiveness (Recall/Precision/F_1/Accuracy) vs. Running time trade-off of different methods on the Gowalla dataset. $|U| = 400, \epsilon = 3.0$.


(a) Recall.


(b) Precision.


(c) F_1 score.

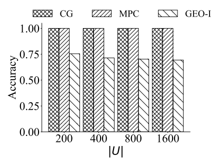
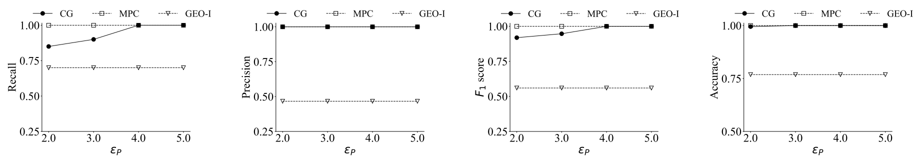

(d) Accuracy.

Fig. 8: Effectiveness (Recall/Precision/F_1/Accuracy) when varying the number of users ($|U|$) on the synthetic dataset. $\epsilon = 4.0$.

when $\epsilon_P = 2.0$, it obtains a similar level as to the one of MPC when a larger value is given.



(a) Recall. (b) Precision. (c) F_1 score. (d) Accuracy.

Fig. 9: Effectiveness (Recall/Precision/F_1/Accuracy) when varying the privacy budget for the patients (ϵ_P) on the Gowalla dataset. $eps = 4.0, |U| = 400$.