Provable Privacy Guarantee for Individual Identities and Locations During Virus Contact Tracing

Tyler Nicewarner
Vanderbilt University

Wei Jiang
Oracle Labs

Aniruddha Gokhale
Vanderbilt University

Dan Lin
Vanderbilt University

Abstract

Infectious disease contact tracing has been an important yet challenging task especially when it comes to meeting the stringent privacy requirements. Although there have been various attempts in this line of research, there are also various limitations in these works regarding the applicable scenarios and efficiency. In this paper, we propose a unique contact tracing system called PREVENT which can prevent any party including the servers from knowing the plain texts of the locations of the people being tracked. Our system is also extremely efficient to provide real-time query services in large-scale datasets that contain millions of locations. The system is built upon a newly designed secret-sharing based architecture that is tightly integrated into space partitioning trees. Our experimental results on both real and synthetic datasets further demonstrate that our system introduces negligible performance overhead compared to contact tracing done on plain texts of locations.

1 Introduction

Infectious diseases have long been a critical threat to people's health. The COVID-19 pandemic has caused severe loss to human lives as well as country's economy. Due to its highly contagious nature, one of the most effective ways to stop the spread of such an infectious disease is to quarantine people who may have been exposed to the virus through effective contact tracing, as pointed out by health experts [12]. Currently, contact tracing is typically performed manually by health care professionals via questions and answers with the patients, which is not only extremely labor intensive, highly subjective, but also prone to errors and misinformation. This is because current contact tracing requires significant human efforts to investigate who have been in close contact with the newly diagnosed patient during the virus incubation period which could be as long as two to three weeks. Most of time it is very hard to identify all the people that need to be quarantined since the patient's daily activities in the past few weeks could be in various places with a large number of people including those who did not directly interact with the patient but contacted items that have been touched by the patient.

In order to ease the human efforts, various automatic contact tracing approaches have been proposed. However, it is still an extremely challenging task in terms of achieving full-spectrum of identity and location privacy preservation for individuals while conducting pervasive data collection and efficient big data analysis. Unfortunately, none of the existing industrial and academic works has provided a satisfactory solution to this problem yet. For example, Google and Apple have rolled out Covid-19 contact tracing apps that use Bluetooth radios to track physical proximity between persons. If someone receives a positive COVID-19 diagnosis, any users who have ever been in the proximity of the patient will be identified or alerted for isolation. However, security experts pointed out a long list of potential flaws in the Google and Apple apps especially the privacy concerns that the apps could reveal the identities of Covid-19 positive users or help advertisers track them [4, 18]. Moreover, there is another critical limitation of these apps. They are not able to find all the people who may have been exposed to the virus which lingers in the air after the patient left. The new science findings pointed out that the COVID-19 virus can flow in the air of a confined space for about 3 hours. That means people who came to the place shortly after the patient left are still in the risk of contracting the virus. Since these people were never in close proximity with the patient, the Bluetooth-based app will not record any information in this case. This problem is somehow mitigated by the recent work that uses QR code scanning to record people who visited the same places possibly during different time intervals [13], but it is limited to places that have set up QR codes and patients who later visited and met other people at places without QR codes (such as sharing a car ride) will not be traceable.

In the state of the art, although there may be some works on seemingly similar topics such as location privacy protection [7], privacy-preserving trajectory querying [11], privacy-preserving trajectory publishing [5], none of them can address

the above privacy protection challenges during the contact tracing. This is because most of the existing location privacy related works do not require users to reveal both real identities and precise locations simultaneously. For example, it is sufficient for the server to provide local weather forecast to an anonymous user who only discloses his/her city but not the exact location; it is sufficient for the server to analyze traffic flows on anonymized trajectories without knowing the owners of the trajectories. However, the virus tracking incurs new challenges on privacy protection since it needs to query on both real identity and exact location information while preserving both identity and location privacy. Otherwise, there is no way to contact the people who may have been exposed to virus.

In this paper, we propose a practical privacy-preserving crowd tracking system, called PREVENT (Privacy pREserving Virus ENcountering Tracking), which will not only be able to identify the people who have been in contact with patient but also provide provable privacy guarantees for both identities and locations. More importantly, our system enables a more complete trajectory recording of participating users without sacrificing their privacy. The PREVENT system consists of three major parties as illustrated in Figure 1: (i) servers which host the system; (ii) subscribers such as organizations, universities, companies, etc. which participate in the contact tracing program; (iii) users whose trajectories are tracked by the system. The specific data flow in the PREVENT system is as follows. An organization (subscriber) subscribes to the contract tracing service (service provider) and asks its employees (users) to install the PREVENT mobile app. The PREVENT system will collect users' locations anonymously from the mobile app through a variety of outdoor and indoor positioning systems that are available. Once a user has been diagnosed, the user or his/her employer can send a contact tracing query to the PREVENT system which will then conduct analysis directly on encrypted user data and broadcast a list of pseudo identities of the people who may need to be quarantined, to all the subscribing organizations. During the whole process, the system will provide the full-spectrum privacy guarantees to users, which has not been achieved by any other existing systems. Specifically, both users' identities and trajectories will be always anonymous to service providers; location information about users including the patients and people who may have been exposed to virus will never be disclosed to their employers; users only receive simple notifications about potential virus exposure but will not receive any information about when/where they may have been in contact with which patients. In summary, our work makes the following unique contributions:

- We design a novel privacy-preserving contact tracing architecture and secret sharing based information management protocols. Our system achieves more strict privacy guarantees for users than any existing approaches. Not any single party in our system will gain information more than they already possess.
- We design highly efficient query algorithms which is able to identify affected people within milliseconds. This is attributed to a unique pyramidal data structure that arranges encrypted user location information at different levels of spatial granularity. Moreover, our query algorithm not only finds the people who are in contact with patient zero, but also considers the transitive effect whereby a person may be exposed to another person who has encountered the patient zero and developed symptoms later on.
- We have implemented a prototype of the proposed system and conducted extensive experiments over various settings. The experimental results demonstrate that our approach introduces very little overhead compared to non-privacy-preserving approaches in terms of data collection, and our approach is as fast as non-privacy-preserving approaches in terms of queries.
- Our system allows for various location gathering methods. These include but are not limited to GPS, Bluetooth, and door swiping systems.

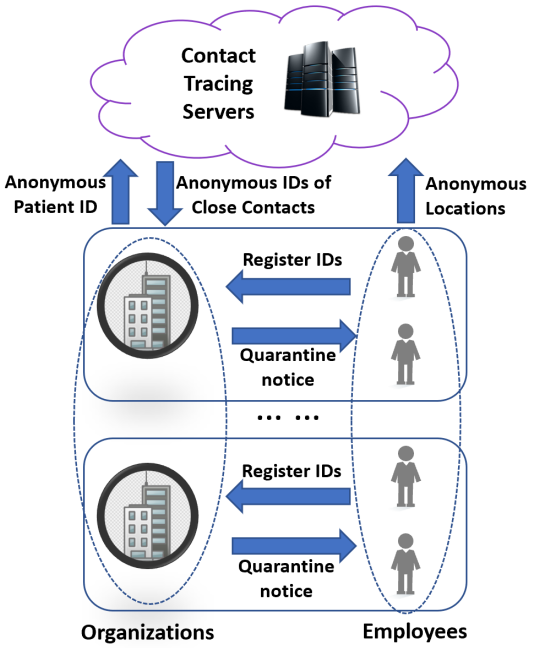

Figure 1: An Overview of the PREVENT System

The rest of the paper is organized as follows. Section 2 reviews the related work on privacy-preserving contact tracing. Section 3 presents our proposed PREVENT system and detailed algorithms. Section 4 analyzes the privacy properties of our system. Section 5 reports experimental results. Finally, Section 6 concludes the paper.

2 Related Works

There have been various works on privacy-preserving contact tracing [18] ever since the pandemic started. However, to the best of our knowledge, none of them handles the same comprehensive scenarios or achieves the same strict security goals as presented in our work, and none of them address the scalability concerns.

Most of the existing works leverage short-range wireless technology such as WiFi and Bluetooth to detect human-to-human contact. Before the pandemic, there have been some works on privacy preserving contact tracing such as the EPIC system proposed by Altuwaiyan et al. [2]. Their solution uses a devices connections to various wireless signals. With the connection information they calculated a weight-based matching score between users. They use a server to store the encrypted signal information and to calculate the score matching calculation. The process is done privately and without disclosing any unnecessary information to the server. However, their approach is inherently computationally expensive and their experiments only tested 5 pairs of people.

Later, Ahmed et al. [1] use secret sharing to help people share encounter IDs over time. They do this by broadcasting one share every minute so that the ID can be reconstructed after k number of shares are received. Once the encounter ID is reconstructed, the encounter is stored in a daily bloom filter located on the users' device. The encounter ID is deleted after insertion into the bloom filter. When someone is diagnosed positively, he/she can upload his/her encounter information to the blockchain. Subsequently, other users can query the blockchain in order to check if they have been in contact with those who were diagnosed positively. This protocol has the limitation that it misses any potential indirect contacts with the virus persisting for a while after an infected person left the area. Also, it requires users to actively check if they have been in contact with the diagnosed patient while our system will automatically notify users. Similarly, Ali and Dyo [3] use Shamirs secret sharing to detect encounters with a beacon using Bluetooth low energy. They have the users send out their ID shares over time just like that in [1]. This approach also has the same limitation that it only tracks people who physically encounter each other.

Trieu et al. [19] propose a solution called Epoine that use secrete sharing and Bluetooth to allow users to exchange a randomly generated "contact token" for the users to store when they are close to each other. For the system to query, the server broadcasts a set of tokens that are sent by the users who are confirmed to be infected. Other users then compare the tokens received from the server to the tokens gathered from contacts to see if contact with the disease was likely. Following a similar idea, Pinkas and Ronen [14] propose a Hashomer system that relies on Bluetooth to detect close contact among users and record the pseudo IDs of encounters in the application. When a user is diagnosed positive, he/she will provide to the health bureau with all the ephemeral IDs that were sent by his application in the last 14 days. The health bureau will then broadcast these IDs mixed with other reported IDs to all the users for them to check if they may be the close contacts. In order to provide users more control of their privacy, Song et al. [17] propose a notion of self sovereign identity which allows individual users to determine when and whether to share their identities when encountering others. All these Bluetooth-based approaches have a common limitation that they only help users who passed by each other to anonymously exchange information, whereas our approach allows for contact tracing of people who do not necessarily come into direct contact with each other but still potentially came into contact with the virus due to visiting the same space within a short time frame. This problem is somewhat mitigated by one recent work that uses the QR code scanning to record people who visited the same places [13]. The approach requires the event owner to set up QR codes beforehand, which means the tracing is limited places that have QR codes. Patients who later visited and met other people at places without QR codes will not be traceable.

There are several approaches which allow privacy preserving queries on entire trajectories rather than just encounters recorded by short-range communication. For example, Kim et al. [9] propose to use functional encryption to encrypt users' trajectories and then perform queries directly on encrypted data. However, their settings will require all the users to use the same encryption key to generate encrypted trajectories which will be stored by the server. This may not be secure enough since an attacker just needs to compromise a single user to decrypt the whole dataset. Reichert el al. [15] propose to apply secure multi-party computations among all users. As it requires all users to participate in SMC to calculate if their locations were ever in the infectious area of others, it is not scalable when there are a large number of users like the city and multi-organization setting in our work. Their work does not present any experimental results either. Most recently, Zhang et al. [21] propose a block-chain based scheme to achieve privacy-preserving contact tracing in 5G-integrated environment. Their system consists of a trusted medical center and fog nodes. Fog nodes are responsible to log the locations of people near them using 5G and store them on a "permissioned blockchain". Users use their phones to upload their encrypted identities when passing by checkpoints. The users are also able to check if their routes have included any potentially dangerous locations by checking the blockchain. In their system, users will not know others' exact location information, but the medical center is to be assumed trusted and has access to know anyone's locations and track any user. This is different from our system as we ensure that no one party, including servers, are able to gain the location information of a user. In our system, organizations or medical centers will only know the list of user names who may be infected but nothing about the places they have visited preserving their

location privacy.

When it comes to privacy preservation, one may also think about homomorphic encryption. However, this technique may not be suitable here due to the following two reasons. First, consider the number of people and the places they will visit during several weeks. The amount of location information to be analyzed is in astoundingly large scale. However, homomorphic encryption incurs high computational overhead [20] and has not been successfully employed for real-time large-scale data set analysis yet. Second, the homomorphic-based system needs a pair of public and private keys. If we allow individuals to encrypt their locations using the public key and let one of the PREVENT servers to keep the private key, the key server could become a single target. Once an attacker compromises the key server, the attacker will be able to decrypt all the users' encrypted trajectory information.

In addition, there have been works on secure cloud data storage and retrieval [10]. However, those approaches are not applicable to contact tracing because they only allow data owners to securely retrieve their own files whereas contact tracing requires to query other people's information.

3 The Proposed PREVENT System

In this section, we present our proposed privacy-preserving contact tracing system, namely PREVENT (Privacy pREserving Virus ENcountering Tracking). Our system can support large-scale data storage and queries for multiple organizations, multiple cities and states.

3.1 Threat Model

There are three parties in the PREVENT system: contact tracing service provider (servers), subscribers (employers), and users (i.e., people being tracked).

- **Employers**: Organizations and companies which subscribe to the service will help maintain a list of their employees' pseudo IDs for the purpose of notification in case one needs to be quarantined.
- **Users**: People being tracked are data providers who will submit their location information at the end of each day to the servers in an encrypted form. The location information can be collected in a variety of means including GPS, Bluetooth, door swiping systems, etc., to record different types of locations including static locations like supermarket and dynamic locations like buses. The detailed location reporting process will be elaborated later.
- **Service Provider**: Majority of data storage and processing tasks are conducted by the service provider which has multiple servers. Following the same security assumption in many recent works of secure multi-party computation [6], each server is assumed to be independent of one another (e.g., located in different commercial clouds), and will not collude unless being compromised by attackers. To achieve such separation of responsibilities in real world scenarios, one server may belong to a government agency whereas the other server may belong to a company that provides such a tracking service.

In our system, we guarantee that no single party except the user him/herself knows the exact locations. The privacy goals with respect to each participating party are summarized as follows.

- **Users are fully anonymous to service providers.** Servers in the PREVENT system will not know the plain texts of users' real identities and locations.
- **Users' trajectories are fully anonymous to their employers.** System subscribers (organizations or employers) will not know any individual's location information, including when and where he/she has been to. For those who need to be quarantined, only their identities are reported to the employers. Users' location information is always kept secret.
- **Peer users do not know each other's anonymous ID or reported locations**: through our system, users even from the same organization will not know each other's anonymous ID and what location information that others have reported.

3.2 System Overview

The essential function of the PREVENT system is to conduct anonymous contact tracing, which is carried out by answering a privacy-preserving location query issued by the subscriber. In particular, upon a person being diagnosed, a contact tracing request can only be issued by the patient's organization upon receiving notification from the patient's health care provider to prevent any misuse of this tracking system by peer users. Each tracing request by each organization will be logged by the servers for future auditing purposes by the law enforcement. This contact tracing request will contain the pseudo IDs of the patient for the servers to look up the patient's encrypted locations in the database. Then, the servers will search other users' encrypted locations to identify who have passed by the proximity of the places visited by the patient within the risky time windows, e.g., a person stopped by the same place within 3 hours after the patient left. After that, the PREVENT system will broadcast the list of pseudo IDs of the people at risk to all the subscribers. The subscribers (employers) can then look up the local database to find the matching real identities of these pseudo IDs and contact them for quarantine. Definition 1 formally defines the query. In Section 3.6. we will elaborate the detailed query protocols.

Definition 1: Privacy-preserving Contact Tracing Query: *Let D be the infectious distance, τ be the infectious time window, and T be the incubation period. Given a patient's tracking ID u_0, the privacy-preserving contact tracing query Q returns a list of anonymous tracking IDs $U = \{u_1, ..., u_k\}$, whereby u_i satisfies the following condition: at least one location of u_i appears within D distance to at least one location of the previous identified close contact u_j during T and $t_{loc_i} - t_{loc_j} \leq \tau$.*

3.3 Privacy-preserving Data Transmission

The ultimate goal of anonymous data transmission is to prevent any party from knowing the user's location information. In other words, the user's organization, the PREVENT system and other peer users will not know the plain texts of the user's locations. In order to prevent these parties from knowing an individual's location data, the individual has only one choice which is to anonymize his/her location information locally before sending it out to the PREVENT system. This privacy requirement imposes a critical challenge on the design of the subsequent contact tracing queries since queries will need to be conducted on fully anonymized user data too. That means we need to design a data anonymization mechanism that not only hides user's information from other parties but also allow other parties to carry out computations directly on such anonymized data.

To simultaneously meet the stringent requirements on privacy protection, anonymous calculation as well as computational efficiency, we propose the following novel data anonymization and storage mechanisms based on the philosophy of "separation of responsibilities". Specifically, the mobile app at the user side will be in charge of two major functions: (i) stay point recording; and (ii) location reporting. Each employee will pre-generate a large number of pseudo IDs for its employees. The employers will also be responsible for storing the mapping between the pseudo IDs and real identities. The servers will be responsible for all the heavy data storage and processing.

At the user side, the mobile app will use various means of location detection to identify the stay point, i.e., the location that the user stays longer than the infectious time window τ. For example, GPS can be used to record the locations where a user lingers longer than τ, such as a supermarket, a restaurant and a shopping mall. Bluetooth technology similar to some existing apps [14] can be used to activate the location recording when the user encounter other people who rode the same car, bus, subway, or airplane. Specifically, once a close contact is detected by the Bluetooth in the user's smart phone, our app will start recording the duration of the encounter. Once the encounter lasts more than the infectious time, the user's current GPS location will be recorded to be submitted to the server at the end of the day. The encounter with the same person during the same time period will only trigger one-time location recording. Similarly, locations collected from the door swiping systems will be recorded by our app for reporting only if the user stays in the room long enough.

The mobile app will employ the additive secret share scheme to split each recorded stay point and corresponding timestamp into secret shares. Secret shares of different stay points will be attached with different pseudo IDs received from their employers. In this way, none of the servers knows about the user's real ID nor the user's exact location. The additive secret share scheme also helps guarantee that unless more than k servers are under control of an attacker (which adds high cost to the attacker), the user's location will be kept confidential. In order to further prevent the servers from knowing the visiting order of locations, each user will not immediately report the current location to the server. Instead, users only need to send the set of location updates once at the end of each day. Since the timestamps of each location is also a piece of secret share, the servers will not be able to discover which location was visited earlier than another. Specifically, at the end of each day, the user's mobile app will encrypt the secret shares of all the stay points along with their pseudo IDs using the servers' public keys and send them to the servers. The secret shares of the same stay point will be sent to different servers. Also, the secret shares of different stay points will use different IP addresses which can be achieved using VPN apps. This would help prevent the servers from correlating multiple location reports to the same user.

Storing the secret shares of users' location information fulfills the first design goal – the privacy protection. It is still not efficient for the subsequent large-scale contact tracing queries. This is because a brute force approach to finding people at risk would be to compare all of the patient's locations with those of all other users', which is obviously time consuming especially when these comparisons need to be performed via secure computation protocols on secret shares. Therefore, we further enhance the user data organization and develop a data filtering stage to significantly narrow down the search space and obtain a much smaller set of candidate location sets for fine-grained analysis.

Our idea is to hierarchically partition the overall space under consideration into grids with equally-sized cells as shown in Figure 2. To ease the calculation, we trim the overall space into a square which has width of W, minimum latitude x_0 and minimum longitude y_0. The number of levels in the space partitioning tree is denoted as H, the width of the grid cell at the i^{th} level is denoted as w_i where the first level is the lowest. The leaf nodes in the tree store secret shares of users' locations while the internal nodes contain secret shares of grid IDs the users are located. The height and widths of the grid cells at each level are known to all parties, thus the mobile app at each user side can automatically calculate which grid cells the user is currently located using Equation 1, where x and y are the latitude and longitude of the user's location. The grid IDs, GIDs, will be split into secret shares and appended

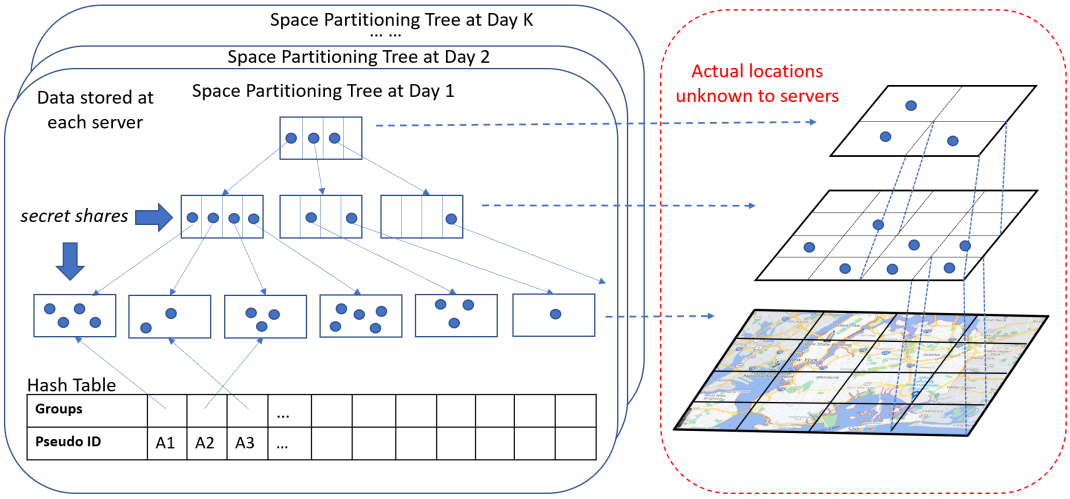

Figure 2: An Overview of Data Structure

to the previously generated secret shares of the exact location and timestamp.

$$GID_i(x,y) = \lfloor \frac{x - x_0}{w_i} \rfloor + \lfloor (\frac{y - y_0}{w_i} - 1) \cdot \frac{W}{w_i} \rfloor \qquad (1)$$

There is a special handling of user's locations within D distance to the border of a grid cell as illustrated in Figure 3. In addition to the previous message, we will create one to three additional messages that contain the new secret shares of the location and the neighboring grid cell IDs. Figure 3 shows an example. The colored circles represent locations of users u_1, u_2 and u_3, respectively. We can see that u_1 is located less than D (infectious distance) to the border of grid cell G_3, which means some users such as u_3 in grid cell G_3 may be within the infectious range of u_1. In order to allow the subsequent contact tracing query to be efficiently conducted within a single grid cell, we will let the servers store u_1's pseudo ID in the grid cell G_3 as well. Specifically, a message that contains newly generated secret shares of u_1's location, grid cell G_3 and its parent grid cell IDs will be sent to the servers in addition to the message for u_1's original grid cell G_4. Similarly, u_3 will also be stored in grid G_4's group at the server side. As for the corner case like u_2, three additional messages will be created to include u_2 in grid cells G_1, G_2 and G_3.

This example also hints that the size of the grid cell at the lowest level should not be too small. In the extreme case when it is smaller than the infectious distance as the overlap from each cell is half the infection distance, all the locations in the cell will need to be included in some neighboring cells. Therefore, we set the cell size at the lowest level to at least 2D (i.e., two times of the infectious distance). It is worth noting that even though the location at the border of edge cell produces a couple of more GIDs, the servers will not be able to know whether the user is located at the edge of a cell based on the total number of GIDs the user sent. This is because the GIDs of all the locations are sent together at the end of day, and different users may visit different numbers of locations. It is indistinguishable to the servers whether a longer list of GIDs is caused simply by users visiting more places or users located at the border of cells.

Later, during the queries, privacy-preserving filtering can be carried out with the aid of the GIDs. The grid cell size is a parameter that can be tuned according to the density of locations to improve the query performance. More details about how to determine the optimal size of cells and how to use GIDs for queries will be elaborated in the next subsection. Algorithm 1 summarizes operations conducted by the user's mobile app.

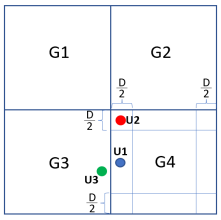

Figure 3: Special Handling of Locations near Borders

3.4 Privacy-preserving Data Storage

At the server side, each server maintains a hash table and a space partitioning tree everyday for latest T days, where T is the incubation period. Such storage can significantly improve the efficiency of the subsequent contact tracing queries as discussed in the next subsections. The hash table stores pseudo IDs, the secret shares of locations, timestamps, and grid IDs along with a pointer to the leaf node of the space

Algorithm 1 Location Data Transmission from a User

Collect a set of pseudo IDs from the server UID^u=$\{UID_1^u, UID_2^u, ..., UID_m^u\}$ (once at the first usage)
Establish a sufficiently large prime number Q
for each stay point loc_k at time t of the day **do**
  Offset Cordinates To Be In Range [0,360]
  $long \leftarrow OffsetCords(loc_k.longitude)$
  $lat \leftarrow OffsetCorts(loc_k.latitude)$

  Calculate N **Secret Shares Of** loc_k
  for $i = 1; i < N; i++$ **do**
    $X_i \leftarrow RandomInteger(0, Q)$
    $Y_i \leftarrow RandomInteger(0, Q)$
  end for
  $X_N \leftarrow (long - \sum_{i=1}^{N-1} X_i) \mod Q$
  $Y_N \leftarrow (lat - \sum_{i=1}^{N-1} Y_i) \mod Q$
  for $i = 1; i \leq N; i++$ **do**
    $l_i \leftarrow (X_i, Y_i)$
  end for

  Calculate The Ids For The Insertion Tree
  for lev = 2; lev $\leq$ H; lev ++ **do**
    Calculate GID_{lev} of loc_k
    Calculate N secret shares of GID_{lev}
    for $i = 1; i < N; i++$ **do**
      $G_{lev}^i \leftarrow RandomInteger(0, Q)$
    end for
    $G_{lev}^N \leftarrow (GID_{lev} - \sum_{i=1}^{N-1} G_{lev}^i) \mod Q$
  end for

  Calculate N **secret shares of timestamp** t
  for $i = 1; i \leq N; i++$ **do**
    Send $\langle UID_k^u, t_i, l_i, G_1^i, G_2^i, ..., G_H^i \rangle$ to Server i
  end for
end for

hierarchy. In the space partitioning tree (left side of Figure 2), users' data with same grid IDs will be grouped together. Specifically, except the leaf nodes that store the pseudo IDs of all the users in the same grid cell, each entry at other levels of the tree only need to store a single pseudo ID of the first user being inserted which is called the representative user. It is important to note that the plain text of the grid ID is hidden from the server. Each server only possesses a piece of secret share of the original grid ID, and hence a single server is not able to obtain the complete grid ID. Also, attributed to the property of the secret sharing scheme, each user's secret share is different from one another even if they are in the same grid cell. Moreover, if a grid cell does not contain a location point, this grid cell will not be stored as an entry in the tree.

Given a user's location update which consists of secret shares of a set of locations (or stay points) visited by the user

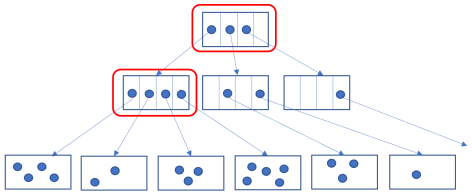

Figure 4: Comparisons Needed for Inserting a Location

in a day, the server will insert each piece of location information as follows. Starting from the root level of the space hierarchy, the servers will conduct a collaborative protocol to compare the root level GID in the newly reported location with that of the representative user at the root level in the current space partitioning tree. If a matching GID is identified, the secure comparison will proceed to the children nodes of this matching GID. The comparison process continues until reaching the leaf node of the space hierarchy. At any level, if there is not any matching GID, a new entry will be created to store the pseudo ID of this user and this user is the representative user of this entry.

The protocol for privacy-preserving comparison of the secret share of the grid ID in the new message (denoted as G_{ui}) and the grid ID of the representative user (denoted as G_i) in the space partitioning tree at Server S_i is outlined in Algorithm 2. First, each server calculates $G_{ui} - G_i$, and stores this difference in d_i. Second, all the servers execute the secure random number generation protocol [8] to generates a random number r. At the end of this protocol, each server only has a share r_i of this random number r but does not know the value of r. Then, each server applies the secure multiplication protocol [8] to derive $v_i = d_i \cdot r_i$, and shares v_i with all the other servers. Finally, every server calculate the sum of v_is, i.e., $\sum_{i=1}^{N} v_i$. If this sum is zero, that means the two grid cell IDs match.

The maximum number of secure comparisons for inserting a new location can be estimated using Equation 2. Specifically, at each level, a new location needs to be compared with all the entries in one node as illustrated in Figure 4. The maximum number of comparisons is equivalent to the sum of the branching factor or width of each level w_i where w is the set of widths for each level. With a tree of depth of H the widths w_0 to w_{H-1} are given as parameters to the system and w_H is the branching factors of the grid cell parents, $\lceil \frac{N}{\prod_{i=0}^{H-1} w_i} \rceil$.

$$C_{insert} = \left\lceil \frac{N}{\prod_{i=0}^{H-1} w_i} \right\rceil + \sum_{i=0}^{H-1} w_i \quad (2)$$

3.5 Optimization of Space Partitioning

We now take a closer look at how we can optimize the insertion process to minimize the maximum number of comparisons required for insertion, C_{insert}. Since the w only con-

Algorithm 2 Secure Comparison For Equality Check

Require: R_j^i is the j^{th} region level on S_i

Require: uR_j^i is the i^{th} share of the j^{th} region level

S_3 **Computes**
$u, v \leftarrow RandomInteger \in [0, Q)$
$w \leftarrow uv$
Generate shares for u, v, w
Send $[u^1, v^1, w^1]$ to S_1
Send $[u^2, v^2, w^2]$ to S_2

S_1 **Computes**
$\alpha^1 \leftarrow R_j^1 - uR_j^1$
$\beta^1 \leftarrow RandomInteger \in [0, Q)$
$\chi^1 \leftarrow \alpha^1 + u^1$
$\gamma^1 \leftarrow \beta^1 + v^1$
Send $[\chi^1, \gamma^1]$ to S_2
$\chi = \chi^1 + \chi^2$
$\gamma = \gamma^1 + \gamma^2$
$d^1 \leftarrow \gamma\chi - \chi v^1 - \gamma u^1 + w^1$
Generate Random Number r^1
Send $d^1 r^1$

S_2 **Computes**
$\alpha^2 \leftarrow R_j^2 - uR_j^2$
$\beta^2 \leftarrow RandomInteger \in [0, Q)$
$\chi^2 \leftarrow \alpha^2 + u^2$
$\gamma^2 \leftarrow \beta^2 + v^2$
Send $[\chi^2, \gamma^2]$ to S_1
$\chi = \chi^1 + \chi^2$
$\gamma = \gamma^1 + \gamma^2$
$d^2 \leftarrow -\chi v^2 - \gamma u^2 + w^2$
Generate Random Number r^2
Send $d^2 r^2$

$D \leftarrow d^1 r^1 + d^2 r^2 \mod Q$
return $D = 0$

sists of positive integer values the term $\sum_{i=0}^{H-1} w_i$ does not effect the ceiling function so we can rewrite Equation 2 as $\left\lceil \frac{N}{\prod_{i=0}^{H-1} w_i} + \sum_{i=0}^{H-1} w_i \right\rceil$. We then drop the ceiling part of the function to focus on the interior section $\frac{N}{\prod_{i=0}^{H-1} w_i} + \sum_{i=0}^{H-1} w_i$. Due to the ceiling function rounding all decimals up to the next integer if we find the minimum of the interior function we are guarenteed to also have the minimum of the entire function. We then look at the domain of positive real numbers, S, and look at the Hessian, H, of our function defined in Equation 3. $H(w)_{i,j}$ can be written as $a \cdot \prod_k \frac{1}{w_k^b}$ for some positive number a and positive integer b. As $x_k > 0$ for any interior point of S, $\frac{1}{x_k^b} > 0$. $H(w)$ is positive for all values in the domain S. Therefore the function is convex over the domain of positive real numbers.

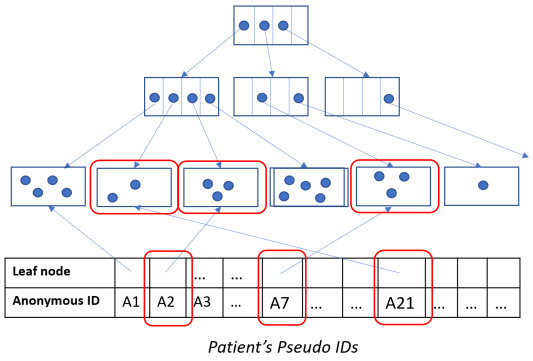

Figure 5: Comparisons Needed for a Query

$$H(w)_{i,j} = \begin{cases} \frac{n}{x_i x_j \prod_k x_k} & i \neq j \\ \frac{2n}{x_i x_j \prod_k x_k} & i = j \end{cases} \tag{3}$$

Since the function is convex any minimum point found will be the global minimum. This allows us to find the minimum of C_{insert} given any N and H using the derivative shown in Equation 4. When solving for each derivative set to 0 we see that $\forall w \in \{w_0 ... w_{H-1}\}$ $w_i = \sqrt[H]{N}$

$$\frac{\partial w_i}{\partial C_{insert}} = 1 - \frac{N}{w_i^2 \prod_{j=0}^{i-1} w_j \prod_{k=i+1}^{H-1} w_k} \tag{4}$$

3.6 Privacy-preserving Multi-generation Contact Tracing Queries

As shown in Definition 1, the goal of a contact tracing query is to identify users who had visited the same places as the patient during the infectious time window. Since each server possesses only a piece of secret share of the user's location data, the contact tracing queries will need to be conducted via a collaborative protocols among multiple servers without leaking users' location information to any server.

The query issuer (e.g., the employer) submits a list of pseudo IDs used by the employee who has been infected to the server. For each received pseudo ID, the server will execute the query as follows. First, the server will retrieve the location information corresponding to the pseudo ID. Then, the servers will securely compare the location visited by the patient with those of other users to see if they are within the infectious region and time window. The pseudo IDs of the identified users will be treated as new query inputs and the query process will be repeated until all possible contacts are identified within the incubation period of the first patient. The final output of the query will be the pseudo IDs of all the people at risk. In what follows, we elaborate how to compare locations in a privacy preserving way.

Assume there are n servers S_1, ..., S_n, each of which contains secret shares of the users' location information during

the length of the incubation period (e.g., last 14 days). Given a patient's pseudo ID UID_p, each server will use the hash table to retrieve the lowest-level leaf node in the space partitioning tree that the patient's location belongs to as shown in Figure 5. As our space partitioning ensures that query results regarding a particular location will be inside the same leaf node that the patient's location resides, the servers just need to compare locations within each retrieved leaf node. The structure of the space partitioning tree is the same across all the servers, i.e., the users are grouped in the same way. The only difference among the space partitioning trees is that different servers store different parts of the secret shares of the same location.

The specific protocol for comparing the secret shares of locations in the same group is depicted in Algorithm 3. Let $l_i(x_i, y_i)$ denote the secret share of the location of the user who is in the same group as the patient at server S_i, and $l_{iq}(x_{iq}, y_{iq})$ denote the querying location (i.e., the patient's location). The goal of this protocol is to check if the user is within the distance of D of the patient. First, each server computes the differences between the secret shares of the user's and the patient's x and y coordinates: $\Delta x_i = x_i - x_{iq}$, $\Delta y_i = y_i - y_{iq}$. Then, all the servers together execute the secure multiplication protocol [8] to compute the square of the differences, i.e., $d_{xi} = (\Delta x_i)^2$, $d_{yi} = (\Delta y_i)^2$. Next, each server sum up d_{xi} and d_{yi} to obtain the share of the square of the Euclidean distance between the user and the patient, denoted as d_i. Finally, all the servers together perform the secure comparison protocol provided in [8] to check if the square of the Euclidean distance is smaller than D^2 (the square of the infectious distance).

The cost of finding close contacts of the patient can be estimated as follows. Suppose that there are currently N_u users' information in the system, and the average number of locations recorded for each user is κ. Assuming all the users' locations are uniformly distributed in the space, each grid cell at the lowest level will contain approximately $\frac{N_u \cdot \kappa}{(\frac{W}{w_1})^2}$ locations. If the average length of user's trajectories is L_u, the average number of grid cells that a user falls in can be estimated as $\frac{l_u}{w_1}$. Since the query will compare the patient's location with the locations in the same grid cell, the query cost can be estimated as the product of the number of grid cells visited by the patient and the number of locations in each grid cell as shown in Equation 5. The cost of the multi-generation query is simply the sum of these individual query cost. From the equation, we can observe that the query cost increases with the grid cell size w_1. This is because larger grid cells contain more locations to be compared. We can also see that the cost reaches highest when w_1 equals the space width W (i.e., no space partitioning). Thus, a relatively small grid cell will benefit the query performance. It is worth noting that the grid size should not be too small to cause extensive special handling cases as discussed in the insertion process.

$$C_{query} = \frac{N_u \cdot \kappa}{(\frac{W}{w_1})^2} \cdot \frac{l_u}{w_1} = \frac{N_u \cdot \kappa \cdot l_u \cdot w_1}{W^2} \tag{5}$$

Algorithm 3 Secure Comparison of Euclidean Distance

Input: (S_i, l_i, l_{iq}, D)
Output: $\Delta l \leq D$
Require: D is the infection distance squared
Require: $l_i \rightarrow (X_i, Y_i)$
Require: $l_{iq} \rightarrow (X_{iq}, Y_{iq})$
1: **for all** S_i **do**
2: $\quad \Delta X_i \leftarrow X_i - X_{iq}$
3: $\quad \Delta Y_i \leftarrow Y_i - Y_{iq}$
4: $\quad d_{xi} \leftarrow SecureMultiplication(\Delta X_i, \Delta X_i)$
5: $\quad d_{yi} \leftarrow SecureMultiplication(\Delta Y_i, \Delta Y_i)$
6: $\quad d_i \leftarrow d_{xi} + d_{yi}$
7: $\quad SecureComparison(d_i, D)$
8: **end for**

4 Security and Privacy Analysis

4.1 Privacy Guarantee

Recall that there are three types of parties in the system: the servers, the organizations (i.e., employers), and the people being tracked. Our system ensures that none of any single party would gain information more than it possesses. That also means an attacker will not be able to gain the location information from attacking any singular party. First, when the server processes the contact tracing query, the server does not know the plain texts of the locations that the patients have been to. After the server found users who may be a close contact of the patients, the server does not know the real identities of these close contacts nor the plain texts of the exact locations of these close contacts. Thus, our system does not reveal locations of patients and close contacts to the server. Second, the server only sends the pseudo IDs of close contacts to employers, so the employers do not know the exact locations of any patient or their employees who have been the close contacts. Thus, our system does not reveal locations to employers. Third, the users who may be in risk would receive a simple message like "you may have been in contact with the virus". From the message, the users cannot tell which patient they were in contact with or when and where they have encountered the patients. Thus, our system does not reveal any patient locations to the close contacts. The following are formal proofs about privacy guarantees achieved by our system.

Theorem 1 *Without any background knowledge, the probability that a server reveals the real identity of a user is no*

more than $\frac{1}{||D_{pi}||\cdot||D_{ri}||}$*, where* D_{pi} *is the domain of all possible pseudo identities,* D_{ri} *is the domain of all possible real identities, and symbol '||D||' denote the number of elements in set D.*

Proof: Each server has only secret shares of users' pseudo IDs. Since each location is associated with a different pseudo ID and IP address, the server will not know which set of location information belongs to the same user from the location reporting process. Thus, given an individual secret share, the server may guess the secret share is corresponding to one of all possible pseudo IDs, i.e., $\frac{1}{||D_{pi}||}$. Given a pseudo ID, the chance for the server to correlate it with the real identity without any background knowledge is $\frac{1}{||D_{ri}||}$. By multiplying these two probabilities, we obtain the probability the server may infer the real identity of a user from the received secret share.

Due to the security of the underlying threshold Shamir secret sharing scheme [16], as long as the number of colluding servers is less than k, these servers cannot derive the original data with the probability higher than that stated in Theorem 1. In addition, the equality and secure comparison protocols have proven to be secure and provided by the well-known MP-SPDZ library [8]. As a result, the servers will not learn anything about the underlying values while executing these protocols. ■

Theorem 2 *Without any background knowledge, the probability that a server knows the smallest grid cell (at the leaf level of the partitioning tree) that a subscriber's location is located is no more than* $\frac{1}{N_g}$*, where* N_g *is the total number of grids at the lowest level of the space partitioning tree.*

Proof: Each server has secret shares of users' location information. Each location secret share is associated with a different pseudo ID and IP address, which prevents the server from correlating multiple locations to the same user. Given an individual location secret share, the server may at most guess this location is in one of N_g possible grid cells, and thus the location disclosure probability is $\frac{1}{N_g}$. ■

Note that this probability is very low as N_g is typically very large. For example, a 3-level partitioning tree with 100 sub-partitions in each partition at each level yields total $100\cdot 100\cdot 100$= 1M cells at the final level, i.e., N_g = 1M. Moreover, the chance the server knows the exact location of a user from the secret share is even lower which will be $\frac{1}{D_l}$, whereby D_l is the domain of all possible locations in the service area.

Theorem 3 *With background knowledge of an outbreak location, the probability that a server correlates the patient's location secret shares with the actual grid cells is no more than* $\frac{q!(4N_v^2-q)!}{(4N_v^2)!}$*, where* N_v *is the number of grid cells the patient's trajectory intercepted, q is the total number of patient's reported locations.*

Proof: If the server has background knowledge that an outbreak of infections occurred at an organization and the patient is from this organization, the server may attempt to infer other locations visited by this patient using this extra background knowledge. Let L_u denote the patient's trajectory length and let w_1 be the length of the smallest grid cell. The number of grid cells intercepted with the patient's trajectory can be estimated as N_v=$\frac{L_u}{w_1}$. Using the grid cell that the organization is located as the center, the patient's trajectory may reach grid cells within the radius of N_v cells. For simplicity, we approximate this total area as a square shape instead of a circle. The number of grid cells in this area will then be $(2N_v)^2 = 4N_v^2$. Since the server does not know the visiting order of the locations, the first randomly picked location secret share could be mapped to one of $4N_v^2$ grid cells; the second location secret share could be one out of $4N_v^2-1$ remaining grid cells; and so on. There are total $\frac{(4N_v^2)!}{q!(4N_v^2-q)!}$ choices of mapping from the secret shares of the patient locations to grid cells. Thus, the probability of inferring the actual grid cells that the patient had passed by is the inverse of the above total choices. ■

We show that the above probability is very small in the real world. Given a grid cell that is 30ft wide, a 5-mile trajectory will intercept about 880 cells, i.e., $N_v = 880$. The possible area the patient may visit could contain $4N_v^2 \approx 3\times10^6$ cells. Even if the patient reported only 3 locations (i.e., $q = 3$), the location inference probability is still as low as $\frac{1}{(3\times10^6)\times(3\times10^6-1)\times(3\times10^6-2)} \approx \frac{1}{27\times10^{18}}$.

Theorem 3 also applies when the server intends to correlate some users' location secret shares to grid cells based on the knowledge of some hot spots such as a place which just held a fair with a large number of attendants. In fact, inferring grid cells using population density of grid cells would be even harder than the previous case when the server knows a patient is from a specific organization. This is because density of grid cells vary throughout a day. The server only receives aggregated density information (i.e., all the users who visited the place throughout the day) since the timestamps are hidden. Also, due to the generation of multiple grid IDs for locations at the border of cells, the density of each cell observed by the server already deviates from the real world density. Further, groups of similar densities are indistinguishable from one another.

Theorem 4 *With or without background knowledge, the probability that an organization can infer its employees' locations more than its background knowledge is nearly 0.*

Proof: The organization stores the mapping between the real identities and the pseudo IDs of its employees. As the location reporting is done directly between the employee's mobile app and the servers but not through the organization, the organization does not have any location related information of its employees. Upon receiving the contact tracing results, the organization will know who are the other employ-

ees in close contact of the diagnosed patient, but still do not know what places they had been to together. Even if the organization has background knowledge of the possible place (e.g., a bar) that the group of infected employees may have been to together, the organization knows no other locations about these employees other than that since the server does not return any location information as the query result.

Also, organizations will not gain any location information by colluding with one another and exchanging their employee's information since they all receive the same query results that contain only pseudo IDs but nothing about locations. ■

Theorem 5 *With or without background knowledge, the probability that a user can infer other users' IDs or locations more than his/her background knowledge is nearly 0.*

Proof: In our Prevent system, peer users are not sharing information with each other, and hence they do not know each other's pseudo ID or location information. Even if an attacker compromises multiple users' mobile apps, the attacker will only know the victims' pseudo IDs and locations, but still nothing about others. ■

4.2 Security and Ethics Problems Discussion

To deploy our system in the real world, it should be integrated with existing security mechanisms for authentication, anti-malware, and network communication security. For example, users' accounts should be validated by their employers and authenticated to their mobile apps. This helps prevent malicious users from creating fake accounts in the servers. Also, the mobile app of our system at the user side should be protected from tampering. As these are common techniques but not our contributions, we will not dive in deeper here.

In addition, our system is designed to encourage more users to participate in contact tracing as they may feel more comfortable of reporting their locations without sacrificing their privacy. However, we do not intend to force users to report locations even if their companies subscribe to the contact tracing services. That means it is possible that some users may turn off the tracking mobile app or simply leave their phones at home. By doing this, they also lose the chance of being notified about potential contact with patients. These human factors are out of the scope of our paper as out goal is to ensure the privacy guarantee of participating users.

5 Experimental Studies

The main goal of the experiments is to evaluate the efficiency of the proposed privacy-preserving data insertion and contact tracing queries. For this, we compare our system with two baseline approaches: (i) the system without any privacy protection (denoted as "NoProtect"), i.e., directly works on plain texts of user data; (ii) the system with privacy protection but without space partitioning tree, denoted as ("NoTree"). All the experiments are conducted on the desktop with Intel Xeon Bronze 3104 1.7GHz CPU, 64GB RAM.

For the experiments, we use datasets derived from the real dataset called GeoLife [22], which contains 17,621 trajectories with a total distance of 1,251,654 kilometers over four years. Each trajectory is represented as a sequence of time-stamped locations with longitude and latitude. These trajectories are originally from 182 users over four years. Since the real dataset is relatively small, we generate synthetic datasets that mimic GeoLife trajectories, in order to test the scalability of our approach. We vary the total number of users from 100K to 500K. Each user's trajectory is generated by picking locations in the real trajectories where people have lingered for the minimum infectious duration. Each trajectory contains maximum 10 locations. We generate one trajectory per user per day for 14 days. This results in maximum 7M locations and 70M locations in the test dataset. It is worth noting that our system can take any types of location as input, and it is not limited to the types of locations which can be collected via various means including GPS, indoor positioning systems, keycard entry systems, etc.

Both the insertion and query performance are evaluated using CPU time. As for the insertion, we record the average insertion time per user when recording all the users' daily trajectories. When testing the query performance, we randomly select 100 users as patients to launch the contact tracing queries, and record the average query time.

5.1 Effect of the Total Number of Trajectories

In the first round of experiments, we vary the total number of trajectories from 1.4M to 7M which are corresponding to 100K to 500K users trajectories in 14 days. The infectious distance is set to 6ft, and the incubation period is 14 days. In this round, we adopt a space partitioning with small grid cells of approximately 30ft by 20ft. Specifically, we first partition the space into 173 region cells, and then further partition each region into 176 cells, which result in a 3-layer space partitioning tree. We compare the performance of our system with a similar system that uses plain text opposed to secret shares to store the data and does not protect privacy.

Figure 6 reports the average insertion cost of the last 100 users to be inserted. It is not surprising to see that our privacy preserving algorithm takes more time than the algorithm that works directly on the plain texts. This is because to achieve privacy preservation, we need to conduct multiple rounds of secure comparison when inserting user's location information. Fortunately, our algorithm is still fast enough to provide real-time services as each user's insertion can be completed still within milliseconds. Moreover, our insertion cost stays nearly constant with the increase of the number of users and trajectories. which demonstrates the scalability of our system.

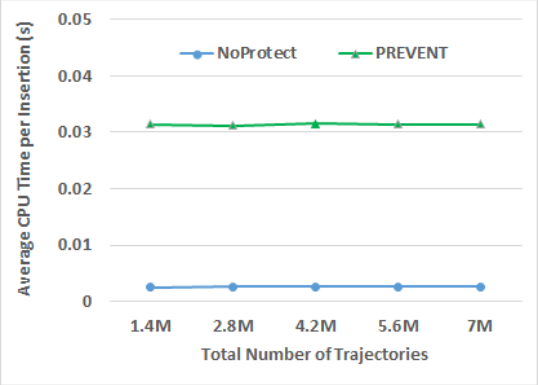

Figure 6: Effect of Total Number of Trajectories on Insertion Performance

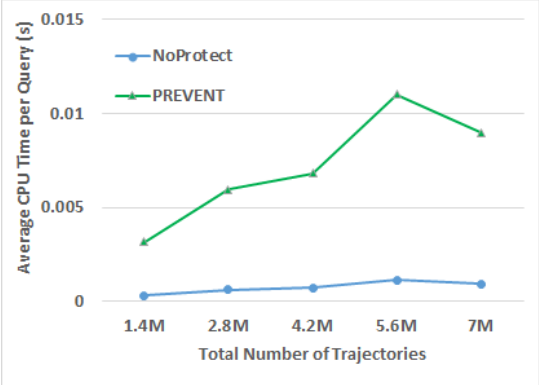

Figure 7: Effect of Total Number of Trajectories on Query Performance

It is attributed to the use of our proposed space partitioning tree. For each insertion, we only need to compare the new user with the single representative user in each grid cell. As long as the space partitioning is the same , i.e., the total number of grid cells stays the same, the insertion performance will not be affected by the total number of trajectories that need to be stored.

Next, we examine the query performance of our algorithm against the baseline approach that has no privacy protection. Figure 7 shows the average query cost of finding the people who were within the infectious distance of a given patient during the incubation period. From the figure, we can observe that the time to perform our secure query is very short, i.e., only a few milliseconds, even thought it is slower than the approach without privacy protection. The overhead of our query algorithm is introduced by the need to conduct secure comparisons between secret shares of users' trajectories. In addition, we also observe that the query cost of both approaches increase with the total number trajectories. The reason is straightforward. In the same space, the more trajectories, the more people may be within the infectious distance of the patient, and hence more comparisons are needed.

5.2 Effect of Space Partitioning

We now take a closer look at the effect of space partitioning by comparing the performance of our approach using small and large grid cells, respectively. Here we use the dataset with 100K users and 1.4M trajectories. The partitioning with small cells are the same as that in the previous experiments whereby each cell is about 30ft by 20ft. The partitioning with large grid cells has the cell size of 300ft by 200ft. Both have three layers. When the small grid cell is used, the entire space is first partitioned into 173 regions and each region is further partitioned into 176 sub-regions. Then, each sub-region contains around 197 small cells. When the large grid cell is used for partitioning, the whole space is first partitioned into 33 regions, and each region is divided into 39 sub-regions. Finally, each sub-regions contains approximately 45 grid cells.

Figure 8 compares the average insertion cost in the following three scenarios: using no space partitioning tree but only one level of large grid cells, partitioning using small grid cells, and partitioning using large grid cells. Observe that the

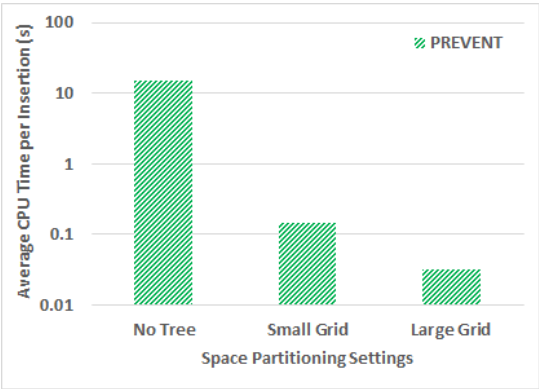

Figure 8: Effect of Space Partitioning on Insertion Performance

insertion cost is highest among all when no space partitioning tree is used, and the insertion cost is lowest when larger grid cells are used. This is because without space partitioning tree, an insertion needs to be compared with the representative user in each grid cell. With the aid of space partitioning tree, an insertion only needs to compare with one grid cell at each level of the space partitioning tree, which significantly reduce the insertion cost. In addition, the larger grid cell also helps reduce the insertion cost. Recall that a user's location near the border of a cell will be included in the neighboring cell as shown in Figure 3. When the size of a grid cell is large, there are fewer such borderline cases, and hence fewer insertions for each user.

Figure 9 shows the corresponding query performance under the same three settings. The query cost is measured using the

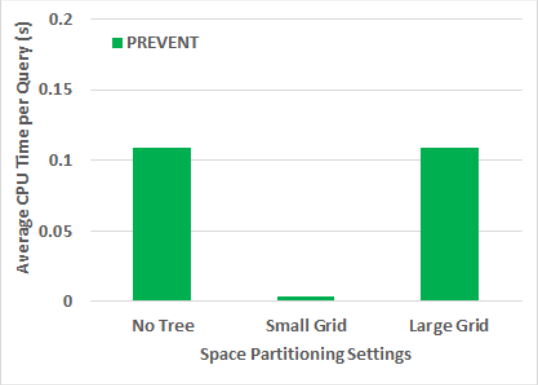

Figure 9: Effect of Space Partitioning on Query Performance

average CPU time of 100 queries. Observe that the query performance is better when the grid cell size is smaller. This is because the query process compares the patient's trajectory with the trajectories in the grid cells that the patient is located. The larger the grid cells, the more candidate trajectories to be compared, and hence results in longer query time. Also, since the query process uses only the hash table but not the space partitioning tree as shown in Figure 5, the query performance of the NoTree version that based on large grid cells is the same as that of our approach using large grid cells.

5.3 Effect of Infectious Distance

This round of experiments evaluates the effect of infectious distance which varies from the typical 6ft distance to a longer distance of 12ft. The dataset used for testing is still the one with 100K users and total 1.4M trajectories. Figure 10 reports the insertion and query cost when large grid cells are used for partitioning. The first observation is that the infectious dis-

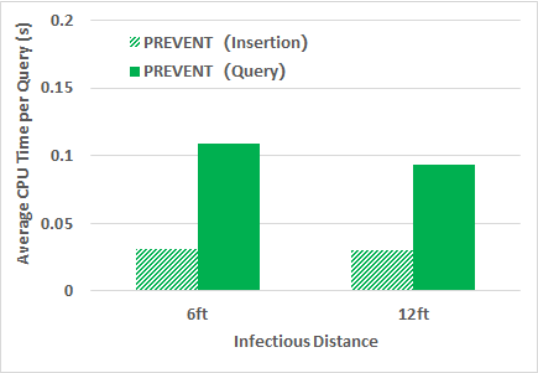

Figure 10: Effect of Infectious Distance

tance does not affect the insertion cost much. This is because the grid cell size is the same. The minor differences in the performance are caused by those data points at the border of cells. When the infectious distance is larger, there are slightly more data points within the infectious distance to the border and need to be included in the neighboring cells. In terms of the query cost, there are not significant differences either. The infectious distance has been doubled while the average query performance is very similar. This is because the query cost is determined mainly by the grid cell size. All the users in the grid cell that the patient has visited will need to be securely compared with the patient's trajectory regardless the length of the infectious distance.

5.4 Effect of Incubation Period

Finally, we study the effect of the incubation period by varying it from 3 days to 14 days. The incubation period only affects the query performance but not the insertion performance since the space partitioning tree structure stays the same. Figure 11 shows the average query cost on the 100K user dataset when using the large grid cell partitioning and 12ft infectious distance. As expected, the longer the incubation period, the higher the query cost. This is because longer incubation period requires the query to compare with trajectories across more days, and hence takes more time.

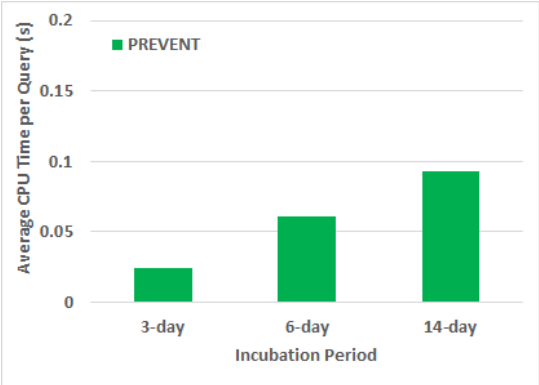

Figure 11: Effect of Incubation Period

6 Conclusion

This paper proposes a unique privacy preserving system, namely PREVENT, for infectious disease contact tracing across multiple organizations. Our system prevents any individual party from knowing the exact locations of the users during the whole process of tracing including location collection and location queries. A unique hierarchical query algorithm has been proposed to ensure real-time performance while offering privacy protection. The experimental results have demonstrated that the proposed system significantly outperforms basic privacy-preserving approaches that do not have the data structure support as that in the PREVENT system, and our system is scalable for handling hundreds of millions of location data.

References

[1] Nadeem Ahmed, Regio A. Michelin, Wanli Xue, Guntur Dharma Putra, Sushmita Ruj, Salil S. Kanhere, and Sanjay Jha. Dimy: Enabling privacy-preserving contact tracing, 2021.

[2] Thamer Altuwaiyan, Mohammad Hadian, and Xiaohui Liang. Epic: Efficient privacy-preserving contact tracing for infection detection. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, 2018.

[3] Vladimir Dyo and Jahangir Ali. Privacy-preserving identity broadcast for contact tracing applications. *2021 Wireless Days (WD)*, Jun 2021.

[4] Anindya Ghose, Beibei Li, Meghanath Macha, Chenshuo Sun, and Natasha Ying Zhang Foutz. Trading privacy for the greater social good: How did america react during covid-19?, 2020.

[5] Sashi Gurung, Dan Lin, Wei Jiang, Ali Hurson, and Rui Zhang. Traffic information publication with privacy preservation. *ACM Transactions on Intelligent Systems and Technology*, 5:1–26, 09 2014.

[6] Jian Kang, Dan Lin, Wei Jiang, and Elisa Bertino. Highly efficient randomized authentication in vanets. *Pervasive and Mobile Computing*, 44:31–44, 2018.

[7] Jian Kang, Doug Steiert, Dan Lin, and Yanjie Fu. Movewithme: Location privacy preservation for smartphone users. *IEEE Transactions on Information Forensics and Security*, PP:1–1, 07 2019.

[8] Marcel Keller. Mp-spdz: A versatile framework for multi-party computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 1575–1590, New York, NY, USA, 2020. Association for Computing Machinery.

[9] Wooil Kim, Hyubjin Lee, and Yon Dohn Chung. Safe contact tracing for covid-19: A method without privacy breach using functional encryption techniques based-on spatio-temporal trajectory data. *PLOS ONE*, 15(12):1–12, 12 2020.

[10] Jingwei Li, Dan Lin, Anna Cinzia Squicciarini, Jin Li, and Chunfu Jia. Towards privacy-preserving storage and retrieval in multiple clouds. *IEEE Transactions on Cloud Computing*, 5(3):499–509, 2017.

[11] Dan Lin, Elisa Bertino, Reynold Cheng, and Sunil Prabhakar. Position transformation: a location privacy protection method for moving objects. *Cyber Center Publications*, 01 2008.

[12] Melika Lotfi, Michael R. Hamblin, and Nima Rezaei. Covid-19: Transmission, prevention, and potential therapeutic opportunities. *Clinica Chimica Acta*, 508:254–266, May 2020.

[13] Wouter Lueks, Seda Gurses, Michael Veale, Edouard Bugnion, Marcel Salathe, Kenneth G. Paterson, and Carmela Troncoso. Crowdnotifier: Decentralized privacy-preserving presence tracing. *Proceedings on Privacy Enhancing Technologies*, 2021(4):350–368, 2021.

[14] Benny Pinkas and Eyal Ronen. Hashomer – privacy-preserving bluetooth based contact tracing scheme for hamagen, 2021.

[15] Leonie Reichert, Samuel Brack, and Björn Scheuermann. Privacy-preserving contact tracing of covid-19 patients. Cryptology ePrint Archive, Report 2020/375, 2020.

[16] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[17] Wenting Song, Razieh Nokhbeh Zaeem, David Liau, Kai Chih Chang, Michael R. Lamison, Manah M. Khalil, and K. Suzanne Barber. Self-sovereign identity and user control for privacy-preserving contact tracing, 2018.

[18] Qiang Tang. Privacy-preserving contact tracing: current solutions and open questions, 2020.

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

[20] Xun Yi, Russell Paulet, and Elisa Bertino. *Homomorphic Encryption*, pages 27–46. Springer International Publishing, Cham, 2014.

[21] Can Zhang, Chang Xu, Kashif Sharif, and Liehuang Zhu. Privacy-preserving contact tracing in 5g-integrated and blockchain-based medical applications. *Computer Standards and Interfaces*, 77:103520, 2021.

[22] Yu Zheng, Hao Fu, Xing Xie, Wei-Ying Ma, and Quannan Li. Geolife gps trajectory dataset - user guide, Oct 2018.