medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-ND 4.0 International license .

Safe contact tracing for COVID-19: A method without privacy breach using functional encryption techniques based-on spatio-temporal trajectory data

Wooil Kim[1], Hyubjin Lee[1], Yon Dohn Chung[1*]

1 Department of Computer Science & Engineering, Korea University, Seoul, Republic of Korea

* ydchung@korea.ac.kr

Abstract

The COVID-19 pandemic has spread all over the globe. In the absence of a vaccine, a small number of countries have managed to control the diffusion of viruses by early detection and early quarantine. South Korea, one of the countries which have kept the epidemics well-controlled, has opened the infected patients' trajectory to the public. Such a reaction has been regarded as an effective method, however, serious privacy breach cases have been issued in South Korea. Furthermore, some suspected contacts have refused to take infection tests because they are afraid of being exposed. To solve this problem, we propose a privacy-preserving contact tracing method based on spatio-temporal trajectory which can be practically used in many quarantine systems. In addition, we develop a system to visualize the contact tracing workflow.

Introduction

As of June 2020, the ongoing pandemic of coronavirus disease 2019, abbreviated as "COVID-19", has caused more than 7 million cases and 0.4 million deaths over 188 countries and territories [1]. Each country has reacted to prevent the epidemic spreads with their ways; Countries in the Schengen area have restricted free movement [2], and Sweden has remained open in an attempt to reach herd immunity [3]. Despite their hard work, only a few countries have kept well-controlled the epidemic spread. Fig. 1 shows the numbers of total cases of COVID-19 over countries which are early detected or currently most highly infected. After the outbreak in Wuhan, China, in December 2019 [4], the virus was first confirmed to have spread to South Korea on 20 January 2020 [5]. By 28 February, more than 2,000 confirmed cases were reported in South Korea [6], and the number of cases exponentially grew for a few days. However, as Fig. 1 shows, the number of total cases in South Korea looks almost not increasing after March, contrary to other countries. South Korea's reaction to the COVID-19 is regarded as a successful method without any lockdowns [7, 8].

The detailed quarantine system of South Korea is described in Fig. 2. If an infected patient is detected, the government puts the patient in quarantine and provides medical care **(1)**. At the same time, they investigate the patient's trajectory based on multiple data sources (i.e. card payment records, mobile GPS data, and self-reports) **(2)**. With the trajectory, the government traces and tests people who have contacted with the infected patient **(4a)**. Considering the undetected contacts, the trajectory is open to the public as an anonymized form **(3, 4b)**. Then the people who might have contacted

NOTE: This preprint reports new research that has not been certified by peer review and should not be used to guide clinical practice.

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint
(which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity.
It is made available under a CC-BY-ND 4.0 International license .

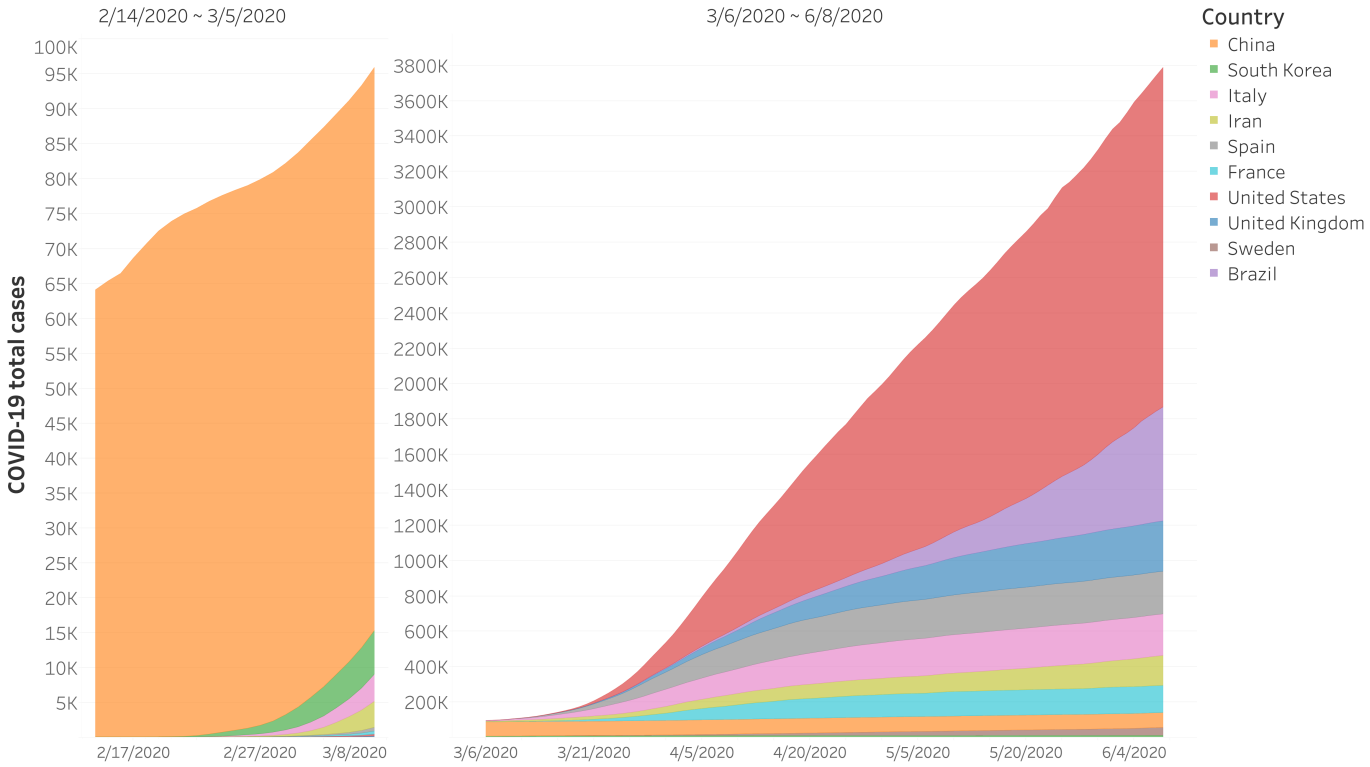

Fig 1. COVID-19 total cases (8. June. 2020) The number of COVID-19 total cases over countries (the early detected countries: China, South Korea, Sweden, and Italy; the most highly infected countries: Iran, Spain, France, United States, United Kingdom, and Brazil).

the patient get tests voluntarily or enforcedly **(5)**. If the contacts are tested as positive **(6a)**, the same loop is repeated and negative contacts are observed and recommended to be self-quarantined **(6b)**. With this system, the government of South Korea reported the lowest number of new cases on 23 March [9]. However, the system also has caused serious privacy problems. Though patients' identifiers (e.g., name) are not open to the public, their privacy has been breached in some extents. The privacy leakage often occurs in the process of disclosing the trajectory to the public. In Fig. 2, when disclosing the trajectory to public **(3)**, tracing the detected contacts **(4a)**, and undetected contacts **(4b)** the privacy problem may occur because the travel route data contain information about individuals' privacy [10].

Commercial [11] and academic [12] solutions have been released for COVID-19 contact tracing. Utilizing these applications and research, people can check whether they have been exposed to the risk of being infected or not. TraceTogether [11] provides secure contact tracing method based on Bluetooth, however, a privacy problem has been issued [13]. Also, He et al. [12] proposed a contact tracing query and its processing method, but the privacy leakage problem is not considered.

No consideration of privacy even decreases the efficiency of epidemic controls because the suspected contacts might refuse to take infection tests because they are afraid of privacy breach. In the absence of vaccine, the importance of contact tracing will increase further. In this paper, we propose a privacy-preserving contact tracing method without privacy breaches. The contributions of this paper are as follows.

- We propose a privacy-preserving contact tracing method based on people's trajectory data.

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-ND 4.0 International license .

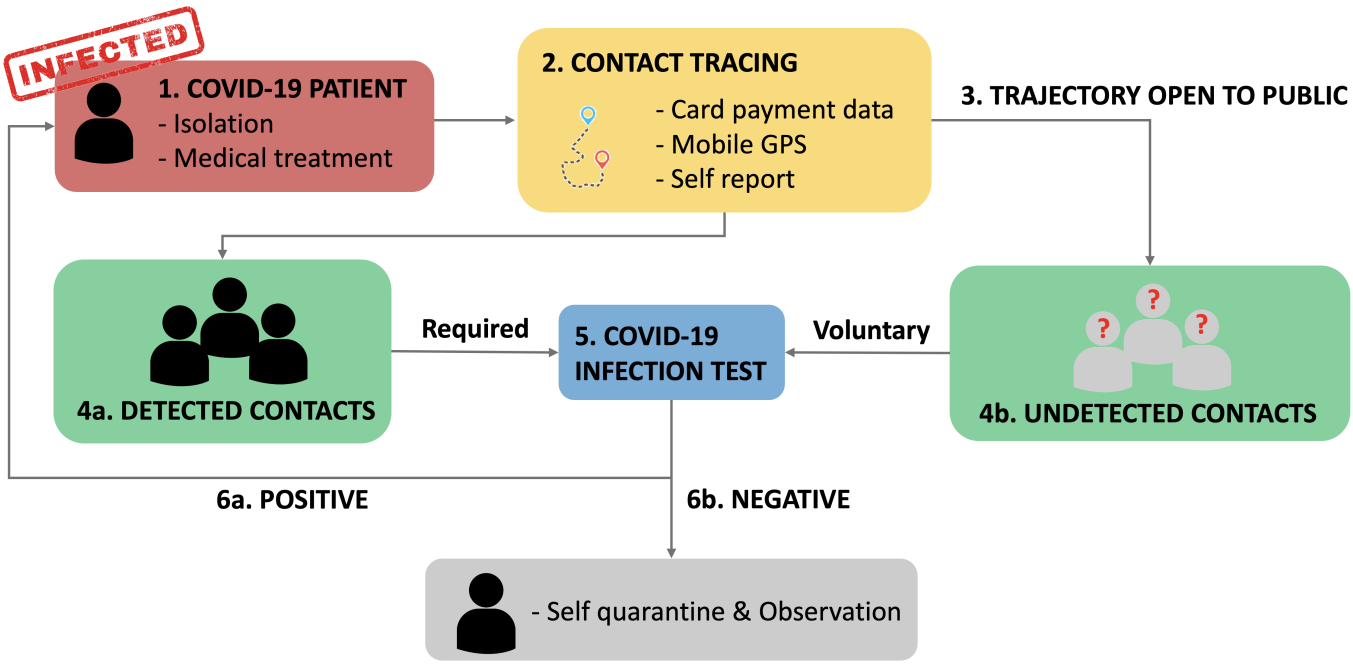

Fig 2. Quarantine workflow of South Korea.

  - We use the credit card payment data as the trajectory source (time and location). The trajectory data (i.e., person ID[1], time and location) are stored in an encrypted form by using a functional encryption technique. Since the trajectory data are encrypted, the time and location privacy of people is preserved. However, the functional encryption scheme allows us to check a person visits the same place at the same time (with a specified time gap) with given confirmed cases' trajectory.

- We enhance the performance of the query processing via parallel processing and optimization techniques.
  - The functional encryption scheme requires a huge processing time, especially during the decryption step. The complexity is known as $O(n^2)$, where n is the domain size. In order to tackle this, we propose a numeric decomposition method which divides numeric values of time and location into digits and encrypts them separately. In addition, we develop a multi-core, parallel processing algorithm with which the processing of contact tracing can be done significantly fast.

- We implement a system to visualize the workflow of contacts tracing process, where the suspected people are detected in real time represented as an infection graph.
  - The GUI system displays the progress of the contact tracing and shows the contamination relation among contacts and confirmed cases into dynamically changing graphs. Also, the probability of contamination is displayed quantitatively for each contact, and the contamination sites are retrieved automatically.

[1]In the paper,we use the card number as the person ID.

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-ND 4.0 International license .

Literature review

Contact tracing methods for COVID-19

Several approaches have been proposed for preventing the spread of COVID-19. Canetti et al. [14] argued that the GPS-based proximity detection methods involve an error inside a building and proposed the method based on Bluetooth. Bluetooth can be a solution for the indoor proximity detection, however, [14] needs an installed application in the devices to store and compute users' data. Singapore's TraceTogether application [11], also works based on Bluetooth. However, [13] showed that the users' infection status and the identifiers are revealed in TraceTogether. Other Bluetooth based methods [15, 16] were proposed considering the privacy. However, all such methods require an additional device or installed application to store and compute the data [14–16], or jeopardize the users' privacy [11]. He et al. propose a new contact tracing method based on trajectory similarity queries [12]. They developed a spatial index (SPT) to reduce I/O and data redundancy. However, the privacy breaches are not considered.

Functional encryption

In a functional encryption system [17], a user is allowed to compute a function F of a ciphertext ct_x (i.e. encrypted version of plain text x) so that the results are available. In other words, a functionality F of ct_x returns the same result of the function of x. Clearly, a functional encryption system guarantees a plaintext x is never disclosed in the process of functionality F. For example, an encrypted dataset and a function with secret key will return only the sum of the dataset.

Our goal is to compute the distance between the two encrypted entities. We adopt the inner product encryption (IPE) for this purpose. The IPE is a special case of functional encryption where both secret keys and ciphertexts are associated with vectors. The combination of a secret key sk_x for a vector x and a ciphertexts ct_y for a vector y reveal only $(x \cdot y)$, not anymore about plaintext x or y. An inner product encryption scheme is function-hiding if the keys and ciphertexts reveal no additional information about both x and y beyond their inner product. The secret key and ciphertext which are presented as vectors, are modified to compute the distance over the encrypted dataset. The details of encryption and decryption for IPE are described in Appendix.

Materials and methods

We define the trajectory as a set of spatio-temporal locations of people. A location may be either a pair of (x, y) coordinate such as longitude and latitude, or a location ID such as the restaurant name.

Definition 1 *A trajectory t^o of a moving object o is a set of pairs of locations and time, where location $l_i^o = (p_i^o, t_i^o), i \in [1, n]$, with p_i^o is a spatial point, s_i^o is a timestamp, and n is the length of trajectory.*

In an encrypted set of trajectories, all spatial points p_i and timestamps s_i are encrypted. Only the objects' IDs are preserved.

Dataset

We use the Gowalla Check-in dataset [18] for conducting experiments and demonstrating our system. The detailed description of the dataset is introduced in

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-ND 4.0 International license .

Table 1. We modified the time period of dataset into one day because the period of two years is not appropriate for contact tracing of COVID-19. The dataset is encrypted with the inner product encryption (IPE) technique [19].

Table 1. Description of Gowalla dataset.

The number of people	107,092
The number of locations	1,280,969
Check-ins	6,442,892
Time period	2 year

Numeric Decomposition (ND)

Table 2 shows decryption time and encryption time of IPE. Although the number of unique values does not effect on the encryption time, the decryption time gets drastically slower. This is due to the fact that the time complexity of IPE decryption (Appendix) is $O(n^2)$. To improve the performance of decryption, we propose a method of decomposing the location ID and visit time into digits. (e.g. 12345 $\rightarrow$ 1, 2, 3, 4, 5) As the numeric decomposition (ND) is applied, the decryption time increases linearly as the number of unique values increases (Table 2).

Table 2. Encryption and decryption time of IPE.

Domain size	Encryption time (seconds)	Decryption time (seconds)	ND applied decryption time (seconds)
10^1	0.1	0.01	0.01
10^2	0.1	0.1	0.02
10^3	0.1	12.6	0.03
10^4	0.1	1595.7	0.04
10^5	0.1	202957.2	0.05

Privacy-preserving contact tracing workflow

We describe our privacy-preserving contact tracing workflow in Fig. 3. First, the trajectories are collected from various sources (e.g. card payment, mobile GPS, and self report). Trajectory data are stored in secured data collectors in an encrypted manner. If a query (i.e., infected person's trajectory data and a time threshold) is given, the

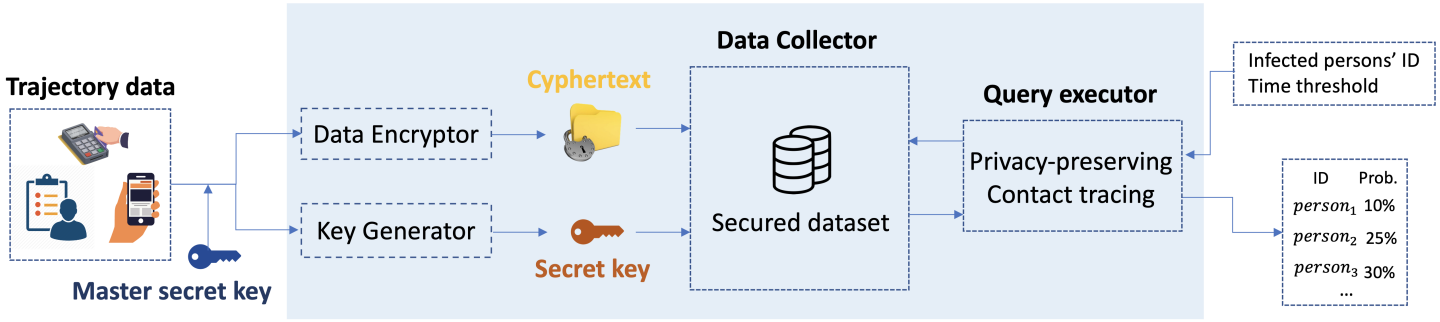

Fig 3. Privacy-preserving contact tracing workflow.

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-ND 4.0 International license .

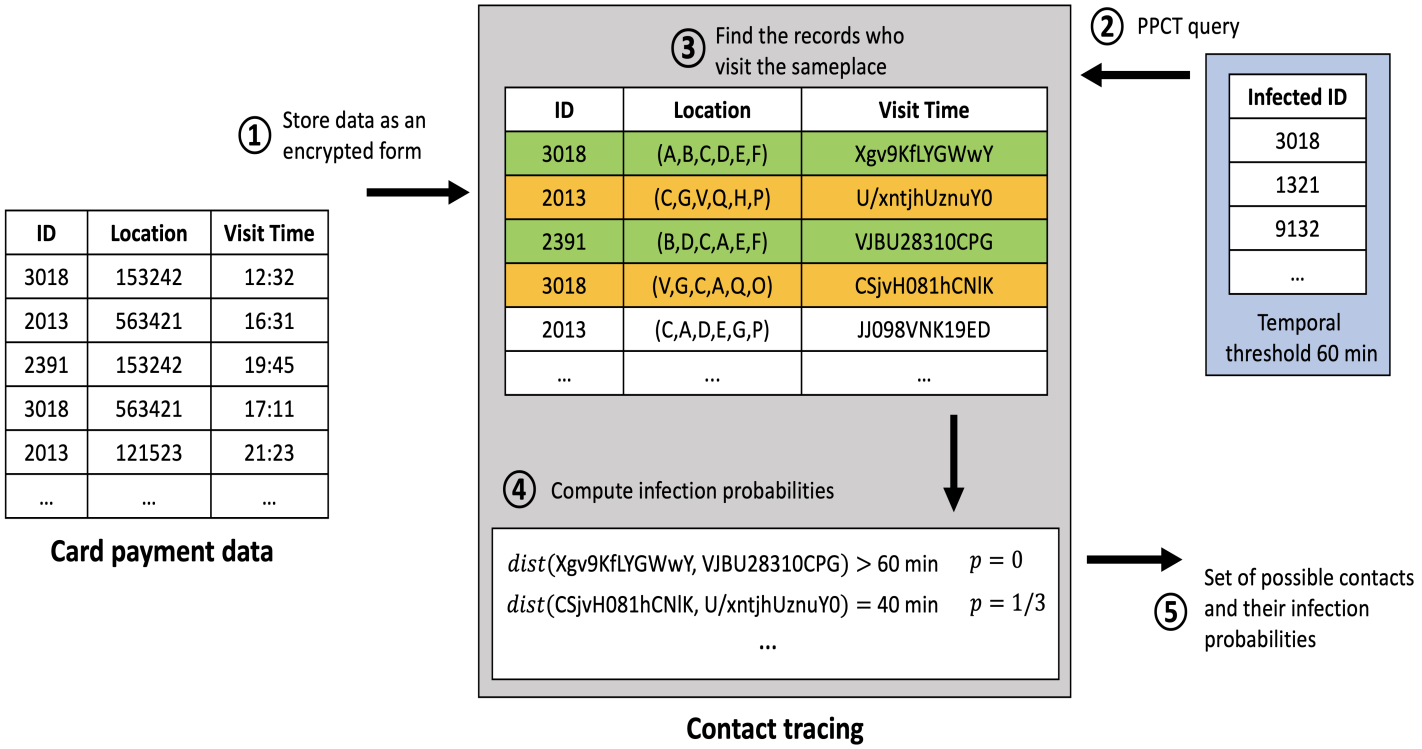

Fig 4. Example of contact tracing with functional encryption techniques The location IDs are decomposed into digits and separately encrypted. The time values are transformed into ordinal numbers and then encrypted. When a set of confirmed IDs is given, the algorithm finds possible contacts by checking the location match and time overlap (within the given threshold 60 min) compared with those of the confirmed cases.

query executor retrieves the possible contacts of infected patients and returns the list of contacts with their infection probabilities.

Privacy-preserving contact tracing (PPCT) method

Given a set of encrypted trajectories $T = \{t_1^1, t_2^1, ..., t_n^m\}$, a set of infected persons $I = \{i_1, i_2, ..., i_k\}$, and time threshold θ, a privacy-preserving contact tracing $PPCT$ retrieves a set of persons $U = \{u_1, u_2, ...\}$ whose infection probability p_{u,i_k}^l is greater than zero. The infect probability of a person u from the infected i_k at the location l_i is defined as Eq. 1

$$\mathrm{p}_{\mathrm{u,i_k}}^{\mathrm{l}} = \begin{cases} min(1, \sum(1 - timegap(u, i_k)/\theta)), & \text{if both } u \text{ and } i_k \text{ visited } l \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The process of privacy-preserving contact tracing is described in Algorithm 1. For efficiency reasons, we check location overlap first, and then compute the time gap.

Fig. 4 describes an example showing how the PPCT algorithm works. First, card payment data are stored as an encrypted form **(1)**. The location is encrypted as a decomposed form. In that, each digit is encrypted as an individual cipher-text. We just notate an alphabet in this example, the real cipher-text is more complicated. Note that the same location 153242 can encrypted into different cipher-texts because the encryption method is not deterministic. When the PPCT query is given as a set of

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint
(which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity.
It is made available under a CC-BY-ND 4.0 International license .

infected patients' IDs and a temporal threshold, the contact tracing algorithm begins **(2)**. Then, the algorithm retrieves the persons who visited the same location with the infected patients **(3)**. In this case, persons 2391 and 2013 have been at the same place 153242 with the infected patient 3018. After that, the algorithm computes the infection probabilities for the overlapped persons **(4)**. The person 2391 was in location 153242 at 19:45 while the infected patient 3018 was in the same location at 12:32. Therefore, their time gap is larger than the temporal threshold (i.e., 60 minutes). The person 2013 was in location 563421 at 16:31 and the infected patient 3018 was in the same location at 17:11. Because their time gap is shorter than the 60 minutes, their infection probability is one third. If the person 2013 and the infected patient 3018 have visited at 563421 several times, the probability will be summed. Surely, the visit time and location are not disclosed in a plain-text form. When the infection probability computation is done for every overlapped visit, the list of possible contacts and their infection probabilities is returned **(5)**.

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-ND 4.0 International license .

Algorithm 1: Privacy-preserving contact tracing PPCT(ids, θ)

Input : List of infected persons' ID ids,
Temporal threshold θ
Output : Set of suspected persons rs
Global : Secretkey list $skxs$, ciphertext list $ctys$, locations, visitorsList
$n \leftarrow$ num of records in $skxs$(or $ctys$)
$m \leftarrow$ domain size of location

```
1  infectedIdx ← []
2  for i ← 0 to n − 1 do
3      if skxs[pid][i] in ids then
4          infectedIdx ← [infectedIdx, i]
5  newLocations, oldLocations ← findLocations(infectedIdx)
6  if len(newLocations) ≠ 0 then
7      for i ← 0 to n − 1 do
8          meetFlag ← False
9          for j ← 0 to len(newLocations) − 1 do
10             if meetFlag == True then
11                 break
12             index ← newLocations[j]
13             if decrypt(skxs[location][index], ctys[location][i], m^2) ≠ 0 then
14                 continue
15             visitorsList[j] ← [visitorsList[j], i]
16             meetFlag ← True
17             time_diff ← decrypt(skxs[time][index], ctys[time][i], θ^2)
18             if time_diff ≠ −1 and time_diff ≤ θ then
19                 add (locations[j], ctys[pid][i], time_diff) to rs
20 for i ← 0 to len(oldLocations) − 1 do
21     if len(oldLocations[i]) == 0 then
22         continue
23     indexes ← oldLocations[i]
24     visitors ← vistorsList[i]
25     for j ← 0 to len(indexes) − 1 do
26         index ← indexes[j]
27         for v ← 0 to len(visitors) − 1 do
28             visitor ← visitors[v]
29             time_diff ← decrypt(skxs[time][index], ctys[time][visitor], θ^2)
30             if time_diff ≠ −1 and time_diff ≤ θ then
31                 add (locations[i], ctys[pid][visitor], time_diff) to rs
32 return rs
```

First, $PPCT$ finds the index of confirmed cases (line 1-4). Then, $PPCT$ checks if a person visits the place where infected patients visited (line 8). If result of decrypt function is 0, it means that secret key and ciphertext are the same (line 13). We store the places where the person visited to skip the decryption next time (line 15). The algorithm finds the person who has contacted with the infected persons at newLocations (line 7-19). Then, $PPCT$ calculates the time gap between the person and the infected (line 17, 29). Lastly, we find the person who has contacted with the infected patients at oldLocations (line 20-31).

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-ND 4.0 International license .

The *findLocations* function is described in Algorithm 2.

Algorithm 2: findLocations(infectedIdx)

Input : infectedIdx
Output : newLocations, oldLocations
Global : Secretkey list *skxs*, ciphertext list *ctys*, locations, visitors
$n \leftarrow$ num of records in *skxs*(or *ctys*)
$m \leftarrow$ domain size of location

```
1  newLocations ← []
2  oldLocations ← [[], [], ...]
3  for i ← 0 to len(infectedIdx) − 1 do
4      index ← infectedIdx[i]
5      isOldLocation ← False
6      for j ← 0 to len(locations) − 1 do
7          if decrypt(skxs[location][index], locations[j], m^2) == then
8              isOldLocation ← True
9              oldLocations[j] ← [oldLocations[j], index]
10             break
11     if isOldLocation == False then
12         locations ← [ctys[location][index], locations]
13         visitorsList ← [[], visitorsList]
14         newLocations ← [index, newLocations]
15         oldLocations ← [[], oldLocations]
16 return newLocations, oldLocations
```

First, we store the index of infected patients who visited place *newLocation* which has never been investigated yet (line 1) and *oldLocations* (line 2). And then, the algorithm checks if the place has been investigated (line 5). If result of decrypt function is 0, it means secret key and ciphertext are the same (line 7). If the infected patients visit investigated place, add index to *oldLocations* (line 8-10). If the infected patients visit not investigated place, add index to *newLocations* and update *locations*, *visitorList*, and *oldLocations* (line 11-15).

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint
(which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity.
It is made available under a CC-BY-ND 4.0 International license .

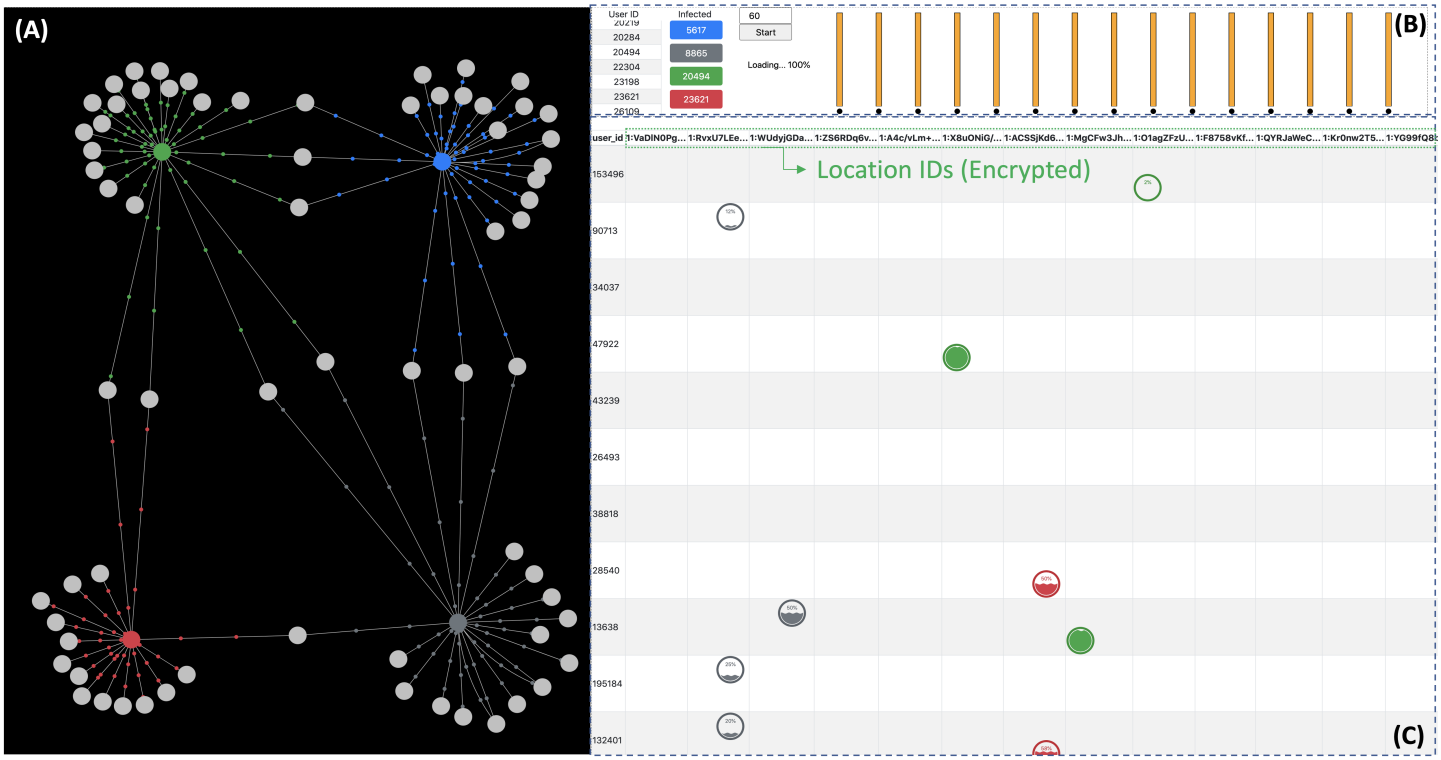

Fig 5. Visualization system. (A) Infection graph; Infected patients and their contacts are depicted as a graph. (B) PPCT executor; The progress of privacy-preserving contact tracing is displayed where a bar represents a piece of CPU core. The progress bars of each core are on the right side. (C) Tracing result table; The infection probability of a person is represented as a wave in a circle.

Visualization system

We implemented a web-based GUI system to visualize the privacy-preserving contact tracing workflow. The system is implemented with python and django [20] for the back-end server, and force-directed graph [21] and d3.js [22] for the front-end side. The overview of our visualization system is depicted in Fig 5. The system is composed of three parts (i.e. (A) Infection graph, (B) PPCT Executor, (C) Tracing result table).

In the infection graph (Fig 5 (A)), the infected patients and their contacts are described as nodes of graph. Infected patients are colored and the contacts are gray-colored. The edges connecting the nodes signify that they have been at the same location in the time-threshold gap. The moving small dots on the edges represent the infection probability. The higher infection probability is, the faster small dots move along the edges. Users might recognize easily the more suspected contacts by the movement of the small dots. The animation is available on the videos in Appendix.

In PPCT executor (Fig 5 (B)), you can choose infected patients and execute the privacy-preserving contact tracing query with a time-threshold. After the query is submitted, the processor redistributes the workload to multi-cores. The processing phase of each core is represented in the progress bars.

In Tracing result table (Fig. 5 (C)), columns denote the locations and rows denote the persons' IDs. As has been stated above, the locations and the time data are encrypted. An infection probability of a person is depicted as a wave in a circle. For example, in the second row and second column of Fig. 5 (C), person 90713 was with the gray-colored infected patient (8865). However, the wave shows only 2% of infection

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint
(which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity.
It is made available under a CC-BY-ND 4.0 International license .

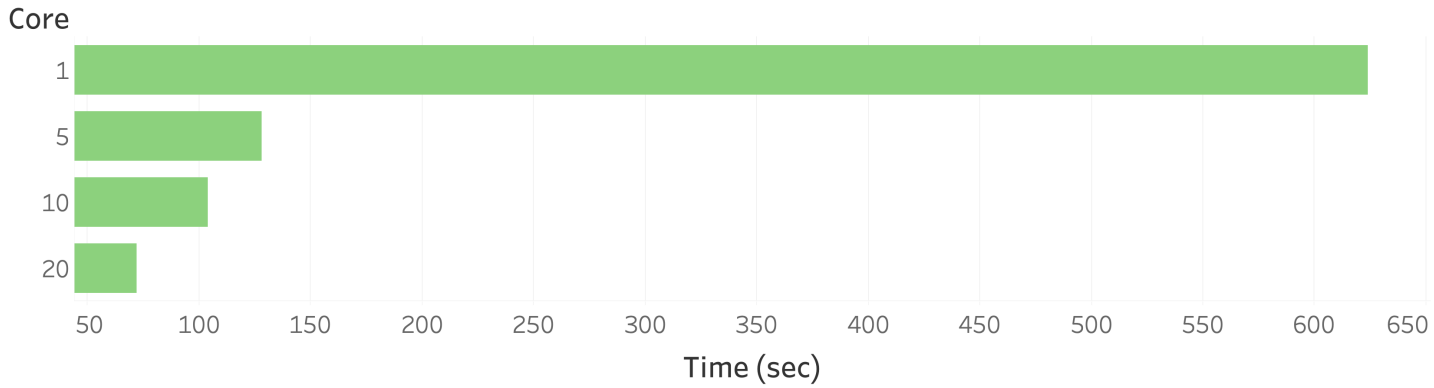

Fig 6. Query processing time varying the number of multi-cores.

probability from the person 90713.

Results

Fig. 6 shows the processing time of privacy-preserving contact tracing on 100,000 records varying the number of CPU cores. Twenty-cores improve the performance about ten times faster. As the main workload of the query processing is independently computing the distance between ciphertext and secret key, the encrypted dataset can be simply divided into the number of cores. Our system is easily applicable to much bigger dataset if enough CPU cores are provided. As the computation of the decryption is independently executable, not only multi-core processing but also distributed computing can be applied to reduce the query processing time. An interesting future work would be a delegatable version of safe contact tracing query to support multi source environment. To handle multi source datasets safely, each dataset should be encrypted with different keys to prevent attacks from other data sources. We hope a very recent work [23] to be a key reasearch for the delegatable safe contact tracing query.

Conclusion

In this paper, we proposed a privacy-preserving contact tracing method for COVID-19 based on a functional encryption technique, where the suspected contacts of infected patients can be retrieved without privacy breaches. To efficiently processing, we introduce optimization techniques and parallel processing. A visualization system is also designed to visualize the workflow of contact tracing.

Supporting information

S1 Video. Video for introducing our visualization. available at `https://youtu.be/5_2TR3l-Fhw`

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-ND 4.0 International license .

S2 Appendix. IPE encryption / decryption algorithm.

Algorithm 3: Setup

Input : Vector size n, prime number p,
two distinct cyclic groups of p order G_1, G_2
Output : master secret key msk

$g_1 \leftarrow$ random element in G_1
$g_2 \leftarrow$ random element in G_2
$B \leftarrow$ random $n * n$ matrix
$detB \leftarrow$ determinant of matrix B
$Bstar \leftarrow detB *$inverse matrix of B
$msk \leftarrow (detB, B, Bstar, g_1, g_2)$
return msk

Setup : set up the master secret key (msk)

Algorithm 4: Keygen

Input : Master secret key msk, plaintext $\vec{x}$, prime number p
Output : Secret key sk_x

$\vec{x} \leftarrow [\vec{x} \cdot \vec{x}, -2\vec{x}, 1]$
$detB, B, Bstar, g_1, g_2 \leftarrow msk$
$n \leftarrow len(\vec{x})$
$\alpha \leftarrow$ random value in the ring of integers modulo p
$k_1 \leftarrow [0, 0, ..., 0]$ k_1's length is n
for $j \leftarrow 0$ **to** $n - 1$ **do**
    $sum \leftarrow 0$
    for $i \leftarrow 0$ **to** $n - 1$ **do**
        $sum \leftarrow sum + (\vec{x}[i] * B[i][j])$
    $k_1[j] \leftarrow \alpha * sum$
for $i \leftarrow 0$ **to** $n - 1$ **do**
    $k_1[i] \leftarrow {g_1}^{k_1[i]}$
$k_2 \leftarrow {g_1}^{\alpha * detB}$
$sk_x \leftarrow (k_1, k_2)$
return sk_x

Keygen : generate secret key (sk_x)
6 - 10 : make $\alpha * \vec{x} \cdot B$
11 - 12 : make $k_1 = g_1^{\alpha * \vec{x} \cdot B}$
13 : make $k_2 = g_1^{\alpha * detB}$

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-ND 4.0 International license .

Algorithm 5: Encryption

Input : Master secret key msk, plaintext $\vec{y}$, prime number p
Output : Ciphertext ct_y

$\vec{y} \leftarrow [1, \vec{y}, \vec{y} \cdot \vec{y}]$
$detB, B, Bstar, g_1, g_2 \leftarrow msk$
$n \leftarrow len(\vec{y})$
$\beta \leftarrow$ random value in the ring of integers modulo p
$c_1 \leftarrow [0, 0, ..., 0]$ c_1's length is n
for $j \leftarrow 0$ **to** $n - 1$ **do**
    $sum \leftarrow 0$
    for $i \leftarrow 0$ **to** $n - 1$ **do**
        $sum \leftarrow sum + (\vec{y}[i] * Bstar[i][j])$
    $c_1[j] \leftarrow \beta * sum$
for $i \leftarrow 0$ **to** $n - 1$ **do**
    $c_1[j] \leftarrow {g_2}^{c_1[i]}$
$c_2 \leftarrow {g_2}^{\beta}$
$ct_y \leftarrow (c_1, c_2)$
return ct_y

Encryption : generate cipherTexts (ct_y)
6 - 10 : make $\beta * \vec{y} \cdot Bstar$
11 - 12 : make $c_1 = g_2^{\beta * \vec{y} \cdot Bstar}$
13 : make $c_2 = g_2^{\beta}$

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-ND 4.0 International license .

Algorithm 6: Decryption

Input : A mapping function e satisfying condition C, secret key sk_x, ciphertext ct_y, maximum inner product value M

Output : dist(sk_x, ct_y)

```
k_1, k_2 ← sk_x
c_1, c_2 ← ct_y
g ← e(k_1, c_1)
h ← e(k_2, c_2)
alpha ← ceiling(M^0.5) + 1
g_inv ← g^-1
t_b ← set
for i ← 0 to alpha do
    t_b[g^(i*alpha)] ← i
    for j ← 0 to alpha do
        s ← h * g_inv^j
        if s in t_b then
            i ← t_b[s]
            return (i * alpha + j)^0.5
return -1
```

3: $g = e(g_1, g_2)^{\alpha\beta * B \cdot Bstar \cdot (\vec{x} \cdot \vec{y})} = e(g_1, g_2)^{\alpha\beta * detB * (\vec{x} \cdot \vec{y})}$

4: $h = e(g_1, g_2)^{\alpha\beta * detB}$

5-14: Baby-step giant-step algorithm for computing the discrete logarithm

Definition 2 ***Condition C*** *Given three distinct cyclic groups G_1, G_2, G_T of prime order p, let $e : G_1 \times G_2 \to G_T$ be a function that maps two elements from G_1 and G_2 onto a target group G_T with following properties*

1. *(Bilinear) $\forall g_1 \in G_1, g_2 \in G_2, a, b \in \mathbb{Z}_p, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$*
2. *(Non-degenerate) $e(g_1, g_2) \neq 1$*

Acknowledgments

This work was partly supported by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2018-0-00269, A research on safe and convenient big data processing methods) and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2017R1A2A2A05069318).

References

1. Wikipedia. COVID-19 pandemic; 2020. https://en.wikipedia.org/wiki/COVID-19_pandemic.
2. Information SV. Schengen Area Crisis: EU States Close Borders as Coronavirus Outbreak Grips Bloc; 2020. https://www.schengenvisainfo.com/news/schengen-area-crisis-eu-states-close-borders-as-coronavirus-outbreak-grip
3. Times NY. 'Life Has to Go On': How Sweden Has Faced the Virus Without a Lockdown; 2020. https://www.nytimes.com/2020/04/28/world/europe/sweden-coronavirus-herd-immunity.html.

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint
(which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity.
It is made available under a CC-BY-ND 4.0 International license .

4. (WHO) WHO. Novel Coronavirus—China; 2020. https://www.who.int/csr/don/12-january-2020-novel-coronavirus-china/en/.

5. Reuters T. 'Like a zombie apocalypse': Residents on edge as coronavirus cases surge in South Korea; 2020. https://www.reuters.com/article/us-china-health-southkorea-cases/like-a-zombie-apocalypse-residents-on-edge-as-coronavirus-cases-surge-in-

6. for Disease Control KC, (CDC) P. The Updates of COVID-19 in Republic of Korea As of 28 February; 2020. https://www.cdc.go.kr/board/board.es?mid=a30402000000&bid=0030.

7. Times. How South Korea's Coronavirus Outbreak Got so Quickly out of Control; 2020. https://time.com/5830594/south-korea-covid19-coronavirus/.

8. Jazeera A. South Korea's coronavirus lessons: Quick, easy tests; monitoring; 2020. https://www.aljazeera.com/news/2020/03/south-korea-coronavirus-lessons-quick-easy-tests-monitoring-2003190114386html.

9. BBC. South Korea reports lowest number of new cases in four weeks; 2020. https://www.bbc.com/news/world-asia-52001837.

10. Douriez M, Doraiswamy H, Freire J, Silva CT. Anonymizing NYC Taxi Data: Does It Matter? In: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA); 2016. p. 140–148.

11. goverment agency S. TraceTogether; 2020. https://www.tracetogether.gov.sg.

12. He H, Li R, Wang R, Bao J, Zheng Y, Li T. Efficient Suspected Infected Crowds Detection Based on Spatio-Temporal Trajectories; 2020.

13. Cho H, Ippolito D, Yu YW. Contact Tracing Mobile Apps for COVID-19: Privacy Considerations and Related Trade-offs; 2020.

14. R. Canetti, A. Trachtenberg, and M. Varia. Anonymous collocation discovery:taming the coronavirus while preserving privacy, 2020;.

15. Apple's and Google's COVID-19 contact tracing technology;. https://tinyurl.com/wfw9ojr.

16. C. Troncoso et al. Decentralized privacy-preserving proximity tracing overview of data protection and security. 2020;. https://github.com/DP-3T/documents.

17. Boneh D, Sahai A, Waters B. Functional Encryption: Definitions and Challenges. In: Ishai Y, editor. Theory of Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg; 2011. p. 253–273.

18. Gowalla check-in dataset; 2020. https://snap.stanford.edu/data/loc-gowalla.html.

19. Kim S, Lewi K, Mandal A, Montgomery H, Roy A, Wu DJ. Function-Hiding Inner Product Encryption Is Practical. In: Catalano D, De Prisco R, editors. Security and Cryptography for Networks. Cham: Springer International Publishing; 2018. p. 544–562.

20. Django;. https://www.djangoproject.com.

medRxiv preprint doi: https://doi.org/10.1101/2020.06.29.20143180; this version posted July 5, 2020. The copyright holder for this preprint
(which was not certified by peer review) is the author/funder, who has granted medRxiv a license to display the preprint in perpetuity.
It is made available under a CC-BY-ND 4.0 International license .

21. D3.js;. `https://github.com/vasturiano/force-graph`.

22. D3.js;. `https://d3js.org`.

23. Li Y, Wang H, Zhao Y. Delegatable Order-Revealing Encryption. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. Asia CCS '19. New York, NY, USA: Association for Computing Machinery; 2019. p. 134–147. Available from: `https://doi.org/10.1145/3321705.3329829`.