A Secure Protocol for Contact Tracing and Hotspots Histogram Computation

by

Chetan Surana Rajender Kumar Surana

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2021 by the
Graduate Supervisory Committee:

Ni Trieu, Chair
Lalitha Sankar
Visar Berisha
Ming Zhao

ARIZONA STATE UNIVERSITY

May 2021

ABSTRACT

Contact tracing has been shown to be effective in limiting the rate of spread of infectious diseases like COVID-19. Several solutions based on the exchange of random, anonymous tokens between users' mobile devices via Bluetooth, or using users' location traces have been proposed and deployed. These solutions require the user device to download the tokens (or traces) of infected users from the server. The user tokens are matched with infected users' tokens to determine an exposure event. These solutions are vulnerable to a range of security and privacy issues, and require large downloads, thus warranting the need for an efficient protocol with strong privacy guarantees. Moreover, these solutions are based solely on proximity between user devices, while COVID-19 can spread from common surfaces as well. Knowledge of areas with a large number of visits by infected users (hotspots) can help inform users to avoid those areas and thereby reduce surface transmission. This thesis proposes a strong secure system for contact tracing and hotspots histogram computation. The contact tracing protocol uses a combination of Bluetooth Low Energy and Global Positioning System (GPS) location data. A novel and deployment-friendly Delegated Private Set Intersection Cardinality protocol is proposed for efficient and secure server aided matching of tokens. Secure aggregation techniques are used to allow the server to learn areas of high risk from location traces of diagnosed users, without revealing any individual user's location history.

DEDICATION

To my parents, and Bruno, whom I love a lot, and the Universe for all the magic along this journey.

ACKNOWLEDGMENTS

I express sincere gratitude for my Thesis chair and faculty advisor, Dr. Ni Trieu, whose guidance, support and valuable mentorship made this thesis possible. Thank you for being the best.

Many thanks to my thesis committee members, Dr. Lalitha Sankar, Dr. Visar Berisha and Dr. Ming Zhao, for their continuous support, guidance and review.

I would like to thank fellow project and lab members Gokulan Vikas Babu, Jiahui Gao and Yitao Chen for their collaboration, encouragement and motivation.

I would like to acknowledge the grant from National Science Foundation (#2031799) and the Google AI for Social Good award for sponsoring the research in this work.

My deepest gratitude to my family and friends, for all their support, encouragement and utmost love every step of the way.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Overview

Our world has been plagued by a pandemic, namely COVID-19, for over a year now, since the first case was reported on November 17th, 2019 [1], [2]. The disease is highly infectious, with a record high number of infections globally [3]. It is an airborne disease that transmits from person to person or from surfaces to persons. Contact tracing (CT), both manual and automated, are effective in limiting the rate of spread of an infectious disease [4]. In CT, people who may have come into contact with an infected person are determined and instructed to self-isolate, to prevent them from transmitting the disease to other people. A large number of CT mobile applications (apps) have been developed and deployed, with most of them based on the exchange of random and anonymous tokens using Bluetooth (BT) [5]–[9]. These are generally decentralized systems that alert users if they may have come in close proximity with other positively diagnosed users (infected users). A high adoption rate is critical for the success of CT apps in helping curb the spread of COVID-19. However, adoption is low as these systems are prone to a host of attacks, like linkage, relay, and replay attacks [10]–[12].

Most CT apps have a service provider(server) in the loop which stores tokens of infected users. User devices periodically query the server and download tokens of infected users. They compute the size of intersection between the set of downloaded tokens and the set of tokens received from other users they were in close contact

with. This involves download of a huge set of data from the server, periodically, hence making it computationally inefficient for client mobile devices.

Moreover, COVID-19 spreads through common surfaces that have been touched by infected users. Informing users to avoid geographical areas where many positively diagnosed users have visited (hotspots) can help lessen the spread of COVID-19 through surface transmission. Decentralized, BT-based systems based on proximity of devices cannot handle this case. GPS based methods that match location traces of users may not be as accurate as BT-based systems, and are vulnerable to dictionary attacks [13]. Hence a secure, efficient, and scalable protocol for CT that considers both contact transmission and surface transmission is essential.

Although several vaccines are available, production is yet to meet demand. Their effectiveness against new mutations of the strain is unknown, and people may still contract the disease, yet suffer a milder case. Thus, there is a need for a framework that is robust against attacks and information leakage, is computationally light and efficient, and allows for valuable insights from anonymous, aggregate data. The study is relevant and advances contact tracing technology to combat highly infectious, airborne diseases and helps prepare for future pandemics.

1.2 Contribution

This thesis proposes, implements and evaluates an end-to-end, secure and efficient Contact Tracing system, with three-fold contributions:

- A secure contact tracing protocol, using a combination of Bluetooth Low Energy (BLE) and GPS, which is efficient, scalable and robust against attacks and information leaks.

- A novel, deployment-friendly protocol with strong privacy guarantees called Delegated Private Set Intersection - Cardinality (DPSI-CA), which allows client devices to offload the computation of matching tokens to detect exposure to an untrusted cloud server.
- A secure and efficient protocol that allows a server to learn geographic infection clusters or hotspots from aggregate user location traces without revealing any individual user's information.

1.3 Outline

Chapter 2 describes related work in the domain of digital contact tracing and server aided private set intersection. Chapter 3 explains relevant background about COVID-19 and contact tracing, as well as defines and elucidates important data structures, cryptographic primitives. Chapter 3 also describes relevant, existing contact tracing frameworks which the proposed work extends and improves. Chapter 4 proposes and explains the secure end to end Contact Tracing framework and the protocols. Chapter 5 provides implementation details and discusses performance of proposed framework. Chapter 6 summarizes the work and discusses future directions for improvement.

Chapter 2

RELATED WORK

2.1 Contact Tracing

Most contact tracing systems based on Bluetooth either assume a trusted central authority or take a distributed approach, depicted in figures 1 and 2. The TraceTogether app [9] launched by the Singapore Government belongs to the former category. In this protocol, the central authority (the government server) registers and stores user details and unique identifiers, and assigns a set of contact tokens to be broadcast at specific times. An infected user I shares all received broadcast tokens with the central authority, who then uses the tokens to identify and follow up with users who have come in contact with I. This system could be misused as a surveillance system, where the central authority can learn graphs of user interaction.

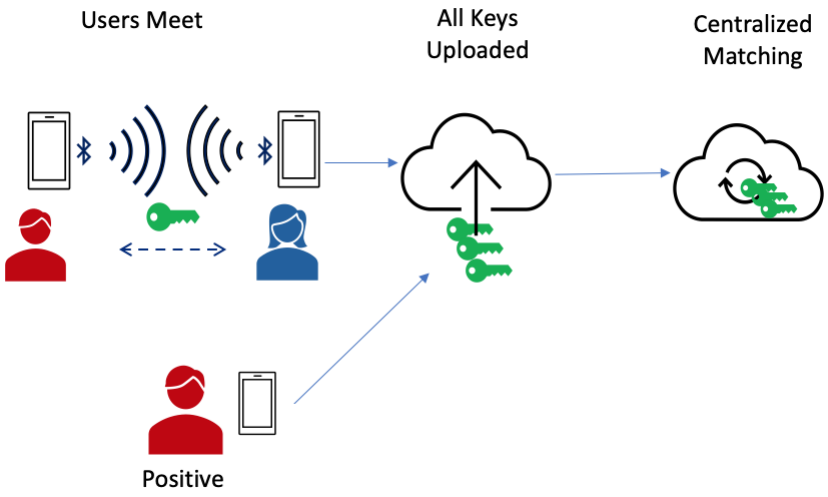

Figure 1. Centralized Bluetooth based Contact Tracing

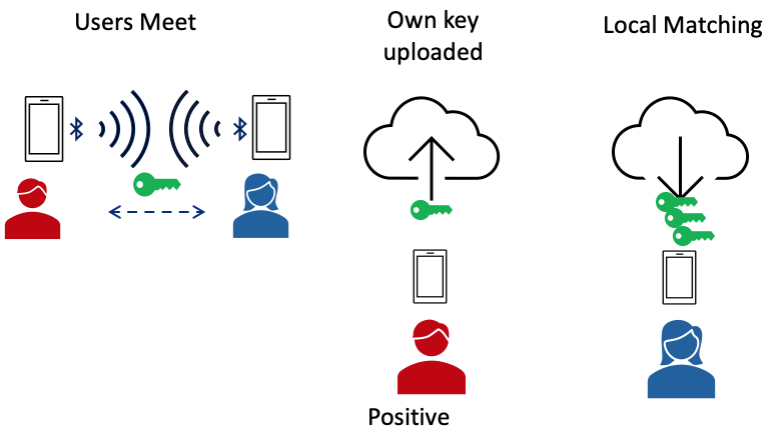

Figure 2. Deentralized Bluetooth based Contact Tracing

Decentralized approaches, including Covid-Watch [6], Decentralized Privacy Preserving Proximity Tracing (DP3T)[7], Temporary Contact Number -Coalition (TCN -Coalition) [8], PACT [14], and the Google/Apple Exposure Notification (GAEN) solution [5], do not trust the central server. These approaches have a common underlying protocol with some differences and can be broadly described as follows. Let's suppose that Alice and Bob are two users of the CT app.

1. Setup: Alice and Bob register and download the app on their devices.
2. Token generation and broadcast: Both Alice and Bob generate contact tokens and broadcast them via Bluetooth Low Energy (BLE). The device rotates the contact token periodically. These tokens/contact tokens are termed as Temporary Contact Numbers (TCN) [8], Ephemeral IDs (EphID)[7], Rotating Proximity Identifiers (RPI)[5], and others.
3. Token exchange: When Alice and Bob are in close proximity, Alice stores Bob's contact token, and Bob stores Alice's contact token. Other metadata including signal strength, duration, timestamp, and others, may be stored with the contact token.

4. Positive diagnosis: Suppose Bob is positively diagnosed. He shares the list of contact tokens broadcast, or the seed used to generate them, with a central, untrusted server that maintains a private database or public list. A health provider may be in the loop to validate or certify Bob's positive diagnosis to the server.
5. Exposure notification: If the server maintains a private database of infected users' tokens, users submit the contact tokens they've encountered to the server. If the server maintains a public list, users download the list. The size of intersection between set of tokens received by the user and set of infected users' tokens gives the number of potential exposure events.

The aforementioned approaches based on this scheme are all susceptible to relay attacks, linkage attacks by users or servers, and also false reporting by users.

- Linkage Attack - Tracking people: In a centralized system, the contact token can be mapped to a long-term psuedonym of the user using a trapdoor. The central authority has access to the trapdoor, and can link all collected or observed ephemeral identifiers. In a decentralized system, the contact tokens are derived from a key k. Diagnosed users submit keys k of several consecutive days. With a public list of contact tokens, an adversary can collect ephemeral identifiers and associated data like time and place of collection, and thereby link k's of a user from different days [15].
- Social graph reconstruction: A determined malicious adversary can learn a part of the social graph in a centralized system. The server can learn the social subgraphs with contacts between diagnosed users and the people they have come in contact with. A determined user can obtain proof of encounter with a diagnosed person in a decentralized system [15].

- Replay and relay attack - Identification of diagnosed users: An adversary, whether an individual, group or organization, can collect contact tokens, from the app or using strong Bluetooth receivers, along with the time and place of collection. In a decentralized system, the tokens of diagnosed users are public. The adversary can use this to a posteriori identify the user that was diagnosed [15].
- False encounter and false reporting: An adversary may install artificial broadcasters, and/or falsely report as positively diagnosed, to increase false positive exposure alerts.

Safe Paths [16], extended to Path Check [17], is one contact tracing approach that is based on GPS location traces of users. The app logs the user's GPS location periodically. The location is quantized to a geographical area using geohash [18]. The app then uses a one way hash function to mask the geohash and timestamp. An infected user's hashes are shared to a central server maintaining a public list. Other devices can download this list and detect an exposure using set intersection. This approach may not be as effective as BT based techniques, and involves a large amount of hashes to be stored locally and downloaded from a server. It is susceptible to dictionary attacks [13], where a one-way deterministic hash used to mask private information can be potentially reversed.

2.2 Private Set Intersection

Private Set Intersection is a multiparty computation (MPC) protocol in which two parties, each holding a dataset, learn the intersection of their sets without revealing any other information. Works on efficient and secure PSI include [19], [20], [21], [22],

[23], [24]. f-PSI refers to protocols for computing a function f on the intersection set. PSI-CA is a special case of f-PSI where the function computed is the size of the intersection set. [20] present a garbled circuit approach for f-PSI based on sort-compare-shuffle construction, which is communication-expensive. [25] propose a Diffie-Hellman Homomorphic encryption approach that has better communication complexity, but is computation intensive and thus exspensive in the mobile setting. Private Set Intersection Cardinality (PSI-CA) [26] refers to a variant of PSI where the two parties only learn the size of the intersection set and nothing else. [26]–[28] propose contact tracing frameworks utilizing PSI protocols. [26] proposes Epione, a decentralized contact tracing system with PSI-CA that is robust against linkage attacks and false reporting, and is compatible with Bluetooth based systems. Their PSI-CA protocol works for asymmetric sets and is linear in the size of the smaller set. Epione uses Diffie Hellman based PSI-CA protocol in the matching process when a user device queries the server to learn any potential exposures. [26] then proposes a PSI-CA protocol extensible to asymmetric sets using Multi -query Keyword - Private Information Retreival (PIR) and two servers. PIR allows a client to query information from one or more servers without the server learning which information was queried. [27] proposes Catalic, another contact tracing framework utilizing delegated PSI-CA protocol, which utilizes oblivious distributed key psuedorandom function (Odk-PRF). Odk-PRF is a distributed version of oblivious PRF (OPRF) protocol. OPRF protocol comprises a sender and receiver. The sender chooses a PRF key k, and receiver learns $F(k, r)$ where r is the receiver's input and F is a psuedorandom function. The sender learns only k, receiver only learns $F(k, r)$ and nothing else respectively. Odk-PRF comprises one sender and multiple receivers, where the input r and output $F(k, r)$ is secret shared among the m receivers. Refer [27] for details on construction of delegated

PSI-CA using Odk-PRf, Cuckoo and Simple hashing scheme, and a pack and unpack message protocol.

[28] takes an approach similar to [26], using Function Secret Sharing (FSS) based on Distributed Point Functions (DPFs) for Private Set Intersection - Cardinality, and a weighted case called PSI with weighted cardinality (PSI-WCA). Two servers hold the keywords X, the infected tokens, and the client holds keywords Y. FSS natively supports secure search of keywords from Y on the set of keywords X. In PSI-WCA, the client obtains sum of weights associated with keywords in the set of intersection between X and Y. While Epione [26] is two-round protocol secure under DDH assumption, [28] proposes a one round protocol with minimal cryptographic assumptions.

Chapter 3

BACKGROUND

3.1 COVID-19 and Contact Tracing

COVID-19 is a highly infectious, airborne disease caused by a coronavirus called SARS-CoV-2 [29]. Its most common symptoms are fever, cough, fatigue and other respiratory issues like loss of taste, smell, sore throat and headache. Severe cases may exhibit symptoms such as shortness of breath, high fever, and persistent pain and pressure in the chest [29]. While most recover, about 15% require hospital treatment and oxygen, and 5% require intensive care. As of April 4th, 2021, over 13 million people worldwide have been diagnosed with the disease, with close to 3 million deaths [3]. Apart from the direct deaths and illness, COVID-19 and the response to handle it indirectly impacted people's mental health and daily lives. It put major strain on the world's economy, with several people losing their jobs and small business struggling to survive [30].

Ways to prevent and slow down transmission include physical distancing measures, wearing masks, keeping areas clean and practicing personal hygiene. Several countries across the globe imposed measures of varying strictness, including lockdowns, mandatory masks, sanitization and so on. Contact tracing (CT) was used with these guidelines to reduce rate of transmission. In CT, the people that may have been in contact with an infected person are determined and alerted of potential exposure. They are advised to isolate themselves to prevent further transmission and may even be required to take a test. CT is effective in reducing transmission of a highly infectious,

airborne disease [4].

Manual contact tracing, however, is tedious and not optimal. It's hard to gather all the people an infected person may have met or visited the same places within a certain time duration after the infected person visited those same places. Following up, alerting them, while at the same time respecting privacy is even more challenging. Digital contact tracing tackles these challenges, leveraging technology to replace or augment manual contact tracing efforts. Section 2.1 describes several digital contact tracing frameworks deployed during the pandemic.

3.2 Threat Model

The protocols in this work are scrutinized under specific security and adversarial models. Consider that multiple parties agree to cooperatively compute a function f, and also agree to share the evaluation result to a particular party. Two classical security models are the colluding and non-colluding models [26]. In a colluding model, a subset of parties may be dishonest and collude during the execution of the protocol. In a non-colluding model, the parties are independent and do not collude.

There are two adversarial model definitions. In the honest but curious or semi-honest model, the parties strictly follow the protocol without deviation, but may attempt to learn extra information from the execution script apart from that intended by the protocol. In the malicious setting, the adversary or dishonest party may attempt any polynomial time strategy such as supplying invalid inputs, deviate and execute different computation, so as to disrupt the protocol and/or leak information. In this work, the non-colluding and semi-honest setting is considered, where the parties are assumed not to collude and follow the protocol.

3.3 Data Structures and Cryptographic Primitives

3.3.1 Geohash

Geohashing [18], [31] is a convenient geocoding system that can encode a location latitude and longitude into a string of letters and digits, with the length of encoding defining the precision. It is a hierarchical spatial data structure that divides geographical areas into grid like buckets. A useful property of a geohash is arbitrary precision, allowing one to gradually remove characters from the end, reducing the length while losing precision. The longer the prefix of geohashes of two locations, the closer they are spatially.

A geohash from GPS coordinates is computed by interleaving two binary strings, one each for the latitude and longitude, with bits recursively splitting grid into intervals. The calculation of a geohash can be elucidated with an example. The interval is between -90 to 90 degrees for latitude, and between -180 to 180 degrees for longitude. Say for a GPS coordinate with latitude 19.5, it lies in the second half of the interval, corresponding to bit 1. Then, 0 is noted for it lies in the first half of the interval 0 to 90, followed by 0 for the interval 0 to 45, 1 for interval 0 to 22.5, and so on recursively, until desired accuracy is reached. The interleaved binary strings for longitude and latitude are represented as letters and digits using the base 32 encoding. The precision for geohash of given length is summarized in table 1. In the implementation of the BT plus GPS protocol proposed in this work, geohash of length 8 is chosen, to accommodate for reasonable accuracy of proximity detection.

Table 1. Geohash precision

Geohash Length	Cell width x Cell Height
1	5,009.4km x 4,992.6km
2	1,252.3km x 624.1km
3	156.5km x 156km
4	39.1km x 19.5km
5	4.9km x 4.9km
6	1.2km x 609.4m
7	152.9m x 152.4m
8	38.2m x 19m
9	4.8m x 4.8m
10	1.2m x 59.5cm
11	14.9cm x 1.9 cm
12	3.7cm x 1.9cm

Cell width and height of grid based on length of geohash. (Units in Kilometer (Km), Meter(m) and Centimeter(cm)

3.3.2 Advanced Encryption Standard (AES)

AES is a symmetric block cipher encryption-decryption algorithm specified as an encryption standard by the U.S. National Institute of Standards and Technology (NIST). AES used in the implementation of proposed work as a one-way hash to mask data. AES is used for hashing in this work over alternate hashing schemes like Secure Hashing Algorithm family (SHA)[32] as it is faster. AES takes a cipher key and plaintext as input and ouputs an encrypted ciphertext. It has a fixed block size of 128 bits, but takes keys of length 128, 192, or 256 bits. Block cipher modes include Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher FeedBack (CFB), Output FeedBack (OFB) and Counter (CTR) [33].

3.3.3 Hash Table Data Structures

- Cuckoo Hashing

  In basic Cuckoo hashing, there are β bins denoted $B[1 \dots \beta]$, a stash, and k random hash functions $h_1, \dots, h_k : \{0,1\}^\star \to [\beta]$. It is a scheme with worst case constant lookup and deletion time, and amortised constant insertion time. On inserting an item, it uses the first hash function. If an item already exists there, the curent item replaces it, and the evicted item is re-inserted using the subsequent hash function. Repeat till the process settles. If there's a cycle, a *rehash* is performed by choosing new hash functions $h_1, \dots, h_k : \{0,1\}^\star \to [\beta]$. The client uses a variant of Cuckoo hashing such that each item $x \in X$ is placed in exactly one of β bins. Using the Cuckoo analysis [34] based on the set size $|X|$, the parameters β, k are chosen so that with high probability $(1 - 2^{-\lambda})$ every bin contains at most one item, and no item has to be placed in the stash during the Cuckoo eviction (.i.e. no stash is required).

- Simple Hashing

  With simple hashing, items in the input set Y are inserted into β bins using the same set of k Cuckoo hash functions (i.e, each item $y \in Y$ appears k times in the hash table). Using a standard ball-and-bin analysis based on k, β, and the input size of client $|X|$, one can deduce an upper bound η such that no bin contains more than η items with high probability $(1 - 2^{-\lambda})$.

3.3.4 Pack and Unpack Message

A pack and unpack message protocol consists two algorithms, pack and unpack. Pack takes a set S of key-value pairs and outputs a representation Π. Unpack takes the representation Π and a key a. It returns the corresponding value associated with the key if and only if $a \in S$, else returns a random value. These algorithms, as stated in [27], are:

- $pack(S) \rightarrow \Pi$: Takes a set S of size n with key-value pairs (a_i, b_i) and outputs a representation Π
- $unpack(\Pi, a) \rightarrow v$: Takes a representation Π, key a and outputs v
- Correctness: If $(a, b) \in S$ and $\Pi \leftarrow pack(S)$ then $(a, unpack(\Pi, a)) \in S$
- Obliviousness: For $pack(a_1, b_1, \ldots, (a_n, b_n))$, The distributions of $unpack(\Pi, a)$ and $unpack(\Pi, a')$ are indistinguishable

[22] present several techniques for pack and unpack constructions, of which two are Polynomial-based construction and Garbled Bloom filter construction. In this work, the polynomial based approach for *pack* and *unpack* is used to make the protocol cryptographically simple and efficient. In polynomial-based construction, $Pack(S)$ carries out polynomial interpolation, returning a polynomial Π of degree $n-1$ over the points $(a_i, b_i), \forall i \in n$. Given a set of points $(x_1, y_1), (x_2, y_2)...(x_n, y_n)$, the Lagrange's interpolating polynomial $P(x)$ of degree $n-1$ is determined as:

$$P(x) = \sum_{j=1}^{n} P_j(x)$$

$$P_j(x) = y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$$

$Unpack(\Pi, a)$ returns the value on evaluating the polynomial at a. It returns $v = P(a)$. Both *pack* and *unpack* satisfy correctness and obliviousness properties.

3.4 Relevant Contact Tracing Deployment

3.4.1 Decentralized Privacy Preserving Proximity Tracing (DP3T)

DP3T is an open protocol for contact tracing using BLE, produced by a core team of researchers and scientists across Europe. [7] proposes three designs - low cost, unlinkable and hybrid design. In all three designs, phones generate frequently changing ephemeral identifiers ($EphID$s). A backend server distributes anonymous exposure information to the app running on each phone [7]. The low cost design is similar to the overview decentralized BT based contact tracing systems described in related work. In low cost design, the $EphID$s of an infected user are linkable from start of contagious window until the time of window, allowing local tracking of infected patients during the past window, using Bluetooth receivers and recording devices to store $EphID$s. In unlinkable design, the list of seeds of infected users is not distributed. The $EphID$s of COVID positive users are hashed and stored in a Cuckoo filter. This prevents linkage of $EphID$s and allows the infected users to redact identifiers that they do not want to disclose.

If a user is diagnosed as positive, they have option to choose set I of epochs for which to reveal their identifiers. Then the device uploads the set $(i, seed_i)$ for all epochs i in I. Requiring $seed_i$ rather than $EphID$ ensures that malicious users cannot claim somebody else's $EphID$ as their own.

The Server creates a new Cuckoo Filter F periodically and for each pair $(i, seed_i)$,

it inserts $H(LEFTMOST128(H(seed_i))||i)$ into F, that is, the hashed string $H(EphID||i)$ where $EphID = LEFTMOST128(H(seed_i))$. All devices download the filter F. Each device checks if any of the EphIDs observed are in F. Using Cuckoo Filter F hides the set of $EphID$s of Covid 19 positive users from general public. However, this design requires more bandwidth and storage.

In the hybrid design, devices generate a random seed $seed_w$ for each time window w and uses these seeds to generate $EphID$s similar to low cost design. It enables user to redact time windows, but has weaker protection against tracking, compared to the unlinkable design. The Google Apple Exposure Notification (GAEN) design is a special case of the hybrid design where w corresponds to 1 day.

3.4.2 Safe Paths

Safe Paths is a GPS location trail based contact tracing solution developed at Massachusetts Institute of Technology, which evolved into the PrivateKit solution. [13] Proximity based solutions using BT are ineffective in that they do not handle the surface and airborne transmission. Coronaviruses can transmit not only through human-to-human transmission but through commonly touched surfaces and environments as well. Safe paths handles this case by checking adjacent time periods as well to accommodate the case when two users may have occupied same space one after the other. The protocol can be said to comprise the following steps:

1. Data collection - User's app logs timestamped GPS points every t minutes
2. Redaction and transformation - All location trails are redacted, transformed and encrypted before upload. Sensitive location details can be redacted through automatic inference and manually by the user. Precise locations are replaced by

a larger geographical areas (geohash) that contain them. Moreover, timestamped GPS points are transformed into "point intervals". These point intervals are encrypted with a one-way hash function.

3. Secure data exchange - User app establishes a secure channel with the designated server and requests point interval data of infected carriers for a specific duration of time for a given region.
4. Risk assessment and notification: The app assesses risk by matching local point interval data with point interval data obtained via (3) and notifies user of any exposure risk.

The GPS points are collected as tuples of latitude, longitude and time. GPS points are mapped to a 3D grid, one with one dimension each for latitude, longitude, and time. These 3D grid cells are called 'point intervals'. These point intervals are obfuscated with a one way hash function. The geographic space is partitioned with geohashes or hexagonal global geospatial indexing system of $H3$ grid. Time is partitioned into intervals, like every 2 or 5 minutes.

An infected user provides redacted, anonymized and hashed point intervals (N_I) to server. Other apps periodically exchange information about their own hashed point intervals with server to detect if their hashed point intervals (N_U) match those shared by diagnosed carriers, using Private Set Intersection protocol.

Geographical space is partitioned into intervals. Two points may be close but fall into different intervals. Thus, a user's app check each of their collected point intervals as well as their adjacent point intervals against diagnosed carriers' point intervals. If H3 hexagonal grid is used, there are 6 surrounding hexagons in grid against each data point. Comparisons required thus become $7N_U x N_I$ rather than $N_U x N_I$

Chapter 4

PROPOSED CONTACT TRACING FRAMEWORK

COVID-19 and other airborne diseases can be curbed by limiting the rate of transmission from person to person. Digital contact tracing efforts utilize technology to identify and alert persons potentially exposed to an infected user. Bluetooth on mobile devices is effective in identifying people in close proximity. Specifically, Bluetooth Low Energy (BLE) is a radio specification for short range communication and well suited for proximity detection due to its accuracy and feasibility. All mobile devices come equipped with Bluetooth functionality, with some rare exceptions. Decentralized BT - based contact tracing systems comprise apps running on users' mobile devices and a service provider (server). Users' apps use BLE to broadcast and receive anonymous tokens. Say two users, Alice and Bob, were in close proximity. Alice stores the token broadcast by Bob and vice versa. This way each user's app stores a list of tokens it has received from other users who were in close proximity. When a user, say Bob, is infected, he uploads the seed used to generate the tokens, or all the tokens, to the server. Other user's download or query the server to determine if they have come in contact with an infected user. Since Alice was in contact with Bob, she will be alerted because the intersection between the set of tokens she has received from other users and the set of tokens of infected users maintained by the server is non-zero.

The potential vulnerabilities associated with solely BT-based contact tracing systems, including linkage and replay attacks, identification of diagnosed users, false reporting and false encounters, are covered in Chapter 2. This chapter proposes a secure,

scalable and efficient contact tracing system with strong privacy guarantees, which is robust against these vulnerabilities. The framework takes a step further to aid in prevention of contacts between users and infected users. It defines a protocol using secure aggregation techniques to identify geographical areas deemed high-risk or hotspots due to several visits by infected users. Users can avoid these areas or exercise increased caution. Moreover, existing systems can easily be extended to incorporate or adopt the secure protocols described.

4.1 End-to-end Framework

An end to end framework for digital contact tracing that is efficient, scalable and has strong privacy guarantees is diagrammed in Figure 3 and explained as follows. The framework comprises an app on users' mobile devices, and three types of untrustworthy servers - backend, cloud and healthcare providers.

1. Setup

During initialization, the cloud server randomly chooses a permutation function $\Pi : [N] \rightarrow [N]$, provides it to the healthcare provider. The healthcare provider randomly chooses N certificates C_i and gives the backend server $\Pi(C_i)$ in order. The healthcare provider randomly chooses a PRG seed c for generating valid certificates, and sends the seed to the backend server, which can locally compute certificate $C_i \leftarrow PRG(c||i)$. The backend server generates a public-private key pair (pk, sk) and sends the public key to every user. Each user/phone u_i randomly chooses a PRG seed s_i which is used to generate the bluetooth tokens. As long as the server's configuration does not change, this phase does not need to be run more than once. Whenever a

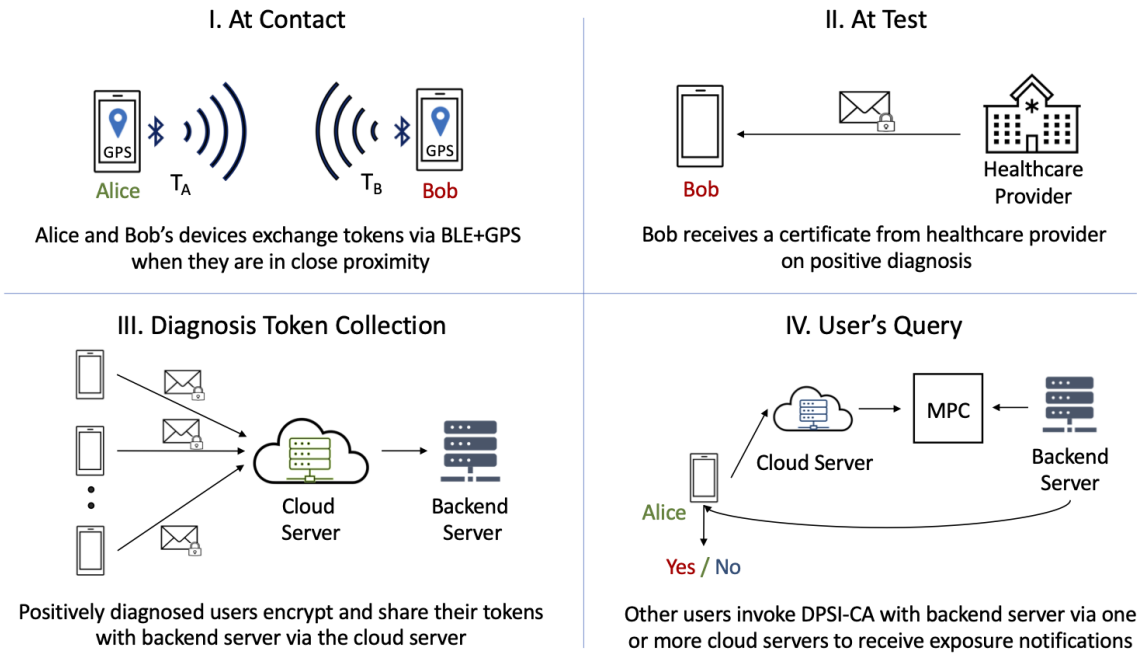

Figure 3. End to End Contact Tracing Framework

(I) Tokens (RPIs) are exchanged when two users are in close proximity. (II) When a user is diagnosed by a healthcare provider, the user receives a certificate which indicates that (s)he tested positive with the disease. (III) the diagnosed user encrypts a pair of their PRG seed or all tokens and the certificate using the public key of the backend server, and sends the encrypted values to the cloud server, who then permutates and transmits them to the backend server. Using its private key, the backend server decrypts the received ciphertexts and obtains a set of pairs including the PRG seed (or all tokens) and associated certificate. The backend server checks whether the certificate is valid using the hospital key. (IV) Each user invokes the DPSI-CA algorithm with the backend server via the cloud server, where the user's input is its received tokens and the server's input is the list of diagnosis tokens. The user learns only whether (or how many) tokens there are in common between the two sets.

new user registers, they only need to generate their own PRG seed and to receive the public key from the backend server.

2. At Contact

The Bluetooth plus GPS protocol described in section 4.2 is used to exchange tokens whenever users are in close proximity. The user can generate the τ tokens to be broadcast by using a PRG as $t_{i,1}||\ldots||t_{i,\tau} = PRG(s_i||d)$, where s_i is the user's secret PRG seed, d is the current day, τ is an upper bound on the number of RPIs needed for that day, and $||$ is string concatenation. Figure (3, I) illustrates the "At contact" phase of token exchange and storage, as explained in section 4.2.

3. At Test

When a user u_i is diagnosed by the healthcare provider, the healthcare provider computes a certificate $C_i \leftarrow PRG(c||i)$ using their own secret PRG seed, and gives it to the user u_i. The certificate validates that this user tested positive for the disease and is used to detect false-positive claims, if any. Note that before adding the user's tokens to the infected tokens database, the backend server checks whether the certificate is valid. If not, the backend server has permission to ask the cloud server to reveal the identify (e.g. the IP address) of this nefarious user. Figure (3, II) presents the "At Test" phase.

4. Token Collection

Figure (3, III) describes the process of collecting diagnosis tokens, which involves the computation and communication of every user, the cloud server, and the backend server. The goal is to have the backend server collect all diagnostic tokens in a privacy-preserving manner. This phase contains three steps as follows:

a) At the beginning of the phase, every i^{th} diagnosed user encrypts their PRG

seed s_i together with their received certificate C_i using the public key pk of the backend server as $Enc(pk, s_i||C_i)$.

b) After receiving the encrypted values from diagnosed users, the cloud server permutes and then forwards them to the backend server.

c) Using its secret key, the backend server decrypts ciphertexts to obtain plaintexts as $s_i||C_i$. First, the backend server verifies whether C_i is valid. This can be done as follows. The backend server uses the PRG seed c of the healthcare provider, generates all possible certificates as $\mathbb{C} = \{C_i \leftarrow PRG(c||i), \forall i \in [N]\}$, and checks whether $C_i \in \mathbb{C}$. If so, the backend server computes all diagnosis tokens as $t_{i,1}|| \ldots ||t_{i,n} = PRG(s_i||d)$, for every d in the infection period, and adds them to the list of diagnosis RPIs $\mathbf{T}$. Otherwise, a false-positive claim is easily detected. A nefarious actor has been caught by communicating with the cloud server and can be held accountable to law.

These steps can be followed when legacy BT based approach is used to exchange tokens. When BT+GPS based approach is used, all hashed tokens combined with location and timestamp rather than just the seed need to be uploaded to the server. The privacy of diagnosed users can be enhanced by allowing every user, including those who have not tested positive yet, to send an encrypted zero value and an "empty" certificate as $Enc(pk, 0||\perp)$ to the cloud server in Step 1. Then, at Step 3, the backend server decrypts ciphertexts and removes all zero values, which belong to non-diagnosed users. By doing so, the cloud server will not know whether a message it receives has come from a diagnosed user. We only require a random subset of the non-diagnosed users, as large as the set of diagnosed users, to be involved. If 1 percent of users test positive, each non-infected user should decide with probability 1/100 to send a zero value to the cloud server.

5. Model Compute and Release

Finally, the backend server holds the uploaded tokens of infected users $\mathbf{T}$ while the i^{th} user holds the received tokens $\widetilde{T}_i$ obtained from the "contact" phase. In the "model compute and release" phase, the user securely compares $\widetilde{T}_i$ with $\mathbf{T}$ by invoking the DPSI-CA protocol. If there is a match, the i^{th} user was in close proximity to a user that has since been diagnosed with the disease. This phase is depicted in (3, IV).

Sections 4.2, 4.3 and 4.4 explain the protocols for BT plus GPS based contact tracing, DPSI-CA for secure matching and the secure aggregation protocol to compute hotspots.

4.2 Bluetooth plus GPS based Contact Tracing

The Bluetooth plus GPS based contact tracing protocol uses the decentralized BT based contact tracing protocol as a baseline and makes it robust against attacks by utilizing GPS location and timestamp data. Linkage attacks exploit the fact that tokens broadcast by user devices can be captured and linked, to reveal the seed used to generate the tokens and thereby track a diagnosed user retroactively when list of infected users' tokens is available. This can be avoided by augmenting the tokens with location and timestamp. The BT plus GPS protocol comprises apps on users' devices equipped with BT and GPS, as well as an untrusted backend server (service provider). The protocol is described as follows:

1. App on user's device continuously broadcasts anonymous tokens T_B that are rotated periodically. The app also listens for any tokens received T_R from other users within a valid range (similar to decentralized BT based CT systems).
2. The app logs the location loc and timestamp t of the user periodically.

3. Let's say Alice and Bob are in close proximity. Alice broadcasts T_{Alice} and Bob broadcasts T_{Bob}. They are at location loc at time t. Both Alice and Bob do the following:

   a) Store $H(T_R + loc + t)$ in list L_R, where T_R is received token, H is a hash function and L_R is the list/table of tokens received. Alice stores $H(T_{Bob} + loc + t)$ and Bob stores $H(T_{Alice} + loc + t)$ in their respective L_R

   b) Store $H(T_B + loc + t)$ in list L_B, where T_B is broadcast token, H is a hash function and L_B is the list/table of tokens broadcast. Alice stores $H(T_{Alice} + loc + t)$ and Bob stores $H(T_{Bob} + loc + t)$ in their respective L_B

4. Suppose Bob is positively diagnosed with the disease. Bob is called an infected user, and his broadcast tokens are called infected tokens. Bob then collects all tokens in his list L_B and uploads to backend server. Note that these tokens are the hash of broadcast BT token, location and timestamp combined.
5. Backend server maintains a public list of infected tokens.
6. Alice invokes DPSI-CA protocol to securely match tokens in her list L_R with the tokens stored by backend server, receiving a count denoting number of potential exposures. If DPSI-CA is not used, Alice downloads the list of infected tokens from server and matches with toekens in L_R. Since token received from Bob is stored in Alice's L_R and with the backend server, Alice receives a non-zero count indicating number of potential exposure events.

When a user is positively diagnosed, the list L_B of tokens that are a hash of broadcast tokens combined with location and timestamp, is required for upload to the backend server. The list L_B may be prepared in two ways:

- Store $H(T_B + loc + t)$ in L_B with periodicity of location logs

- $\{loc, t\}$ entries in a separate table. Compute hash of T_B and $\{loc, t\}$ entries and prepare L_B only when user is positively diagnosed

The list of tokens broadcast, received and infected tokens can be maintained for a certain time period and then deleted, depending on the infectious period of the pathogen.

Security Discussion. The fact that location and timestamp features are incorporated along with the tokens makes it impossible for an adversary, whether the untrusted server or external, to capture and link broadcast BT tokens and attempt to track an infected user. Replay attacks by attempting to rebroadcast a captured BT token at another location to cause false exposure events are avoided as well, since the location mismatch would result in an entirely different token that would not be uploaded to the backend server.

4.3 Delegated Private Set Intersection - Cardinality (DPSI-CA)

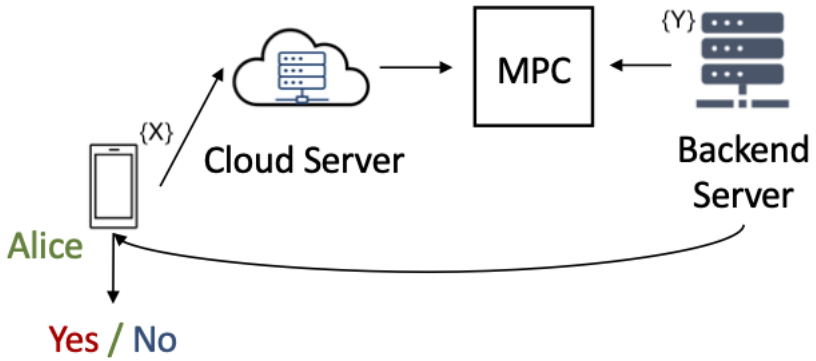

Figure 4. Delegated Private Set Intersection Cardinality (DPSI-CA) Overview

Decentralized BT-based systems require user devices to download list of infected

tokens. The cost and computation analysis in [26] justifies offloading and delegating the matching process from user device to another untrusted server. The DPSI-CA protocol proposed has the following advantages:

- Offloads and delegates matching of tokens, reducing computation cost and avoids large downloads on user device
- Obviates need for a publicly available list of infected tokens, thus removing vulnerabilities such as tracking, identification and determination of social graph
- Compatible with legacy BT-based systems, and BT+GPS protocol described in section 4.1

The DPSI-CA protocol, outlined in figure 5, is a two-server protocol that computes size of intersection set between the dataset of items X held by one party and the dataset of items Y held by another party by using an untrusted server to delegate computation and communication cost from one party. The participants, diagrammed in figure 4, are client (with set X), backend server (with set Y) and cloud server. The client learns $|X \cap Y|$, while backend server learns nothing about X, and the cloud server learns nothing about both X and Y. The complete construction of the protocol is described in 5. The essence of the protocol is a shared secret seed s from which both client and server can generate secret values s_i. Client receives a secret value s_i if and only if $x_i == y_i, \exists y_i \in Y$. The cloud server inserts items from X into a Cuckoo hash table with β bins such that in each bin there is at most one item. The backend server inserts items from Y into a Simple hashing table with β bins using the same hash functions used by the Cuckoo hash table, such that an item is placed in each bin resulting from each hash function used. The backend server uses the polynomial based construction to *pack* the secret values. It generates polynomials for each bin and returns the polynomial coefficients to cloud server. The cloud server obtains the

secret value or a random value on evaluating for each item from client set, and returns all results of evaluation to the client. The client learns the size of intersection by counting the number of correct secret values received.

Security Discussion. From the client's view, they receive a set of values, of which some are secret values generated from the secret seed, and the others a random. Since the client does not know how the cloud server hashes tokens uploaded, it cannot learn which bin, and by extension token, matched with a token held by the server. Hence the client learns only the size of intersection set by counting the number of correct secret values received and nothing else.

The cloud server receives encrypted tokens from the client and polynomial coefficients from the backend server. The cloud server cannot learn or associate client tokens. The backend server pads its bins with a certain number of dummy tokens. Thus the cloud server cannot infer tokens or the count of tokens held by backend server in corresponding bins from the polynomial coefficients. The evaluation of the polynomials on tokens provided by the client appear uniformly random. The backend server obtains tokens from infected users and generates polynomial coefficients after hashing them into Simple Hash Table bins. It learns nothing about client tokens and thus cannot learn a social graph of interaction between users.

4.4 Hotspots Histogram Computation

The hotspots histogram computation protocol determines geographical areas which are visited at least a threshold number of times by infected users. With knowledge of such hotspots, users can avoid such areas, thereby assisting in limiting the spread through exposure prevention. The complete protocol specification is described in

PARAMETERS:

- Set size n and N.
- A client $\mathcal{C}$, a backend server $\mathcal{S}$, and a cloud server $\mathcal{H}$
- One-way hash functions $H : \{0,1\}^\star \to \{0,1\}^\star$, and Cuckoo and Simple hashing scheme described in 3.3.3.
- *pack*() and *unpack*() functions using polynomial interpolation and evaluation described in 3.3.4
- An encryption scheme $Enc(data, key)$ where *data* is the data to be encrypted and *key* is the encryption key.

INPUTS:

- Client $\mathcal{C}$ has input $X = \{x_1, \ldots, x_n\}$
- Backend server $\mathcal{S}$ has input $Y = \{y_1, \ldots, y_N\}$
- Cloud server $\mathcal{H}$ has no input.

PROTOCOL:

I. **Backend server and client setup**
   - The client and backend server share a secret key k and seed s, which is mapped by the client *id*.
   - F_k is a shared function between cloud and backend server. Secret values s_i can be generated by seeding a Psuedo-random generator (PRG) with the seed s.

II. **Tokens distribution phase**
   - Client device computes $x'_i \leftarrow Enc(x_i, k), \forall x_i \in X$, and sends X', id to the cloud server $\mathcal{H}$.

III. **Cloud server computation phase**
   - Cloud server inserts the items from X' into a Cuckoo Hash Table CHT with β bins, such that each bin b has no more than one item.
   - Cloud server requests polynomial coefficients for each bin from backend server, while sharing the client *id* with backend server.

IV. **Backend server computation phase**
   - Backend server computes $y'_i \leftarrow Enc(Y_i, k), \forall y_i \in Y$ by obtaining k associated with client *id* and inserts items from Y into Simple Hash Table SHT
   - Backend server generates the polynomials per bin b as follows:
     1. The bin is padded with random items such that all bins have m items
     2. The interpolating points for the polynomial P_b are $\{(y'_i, F_K(y'_i) + s_b)\}$ for $i \in [1, \ldots, m]$
     3. Backend server returns the coefficients of polynomials P_b for each bin to cloud server

V. **Cloud server evaluation phase**
   - For each item x'_b in bin b, cloud server determines $P_b(x'_b)$ and gets the result $r_b = P_b(x'_b) - F_K(x'_b)$. $r_b = s_b$ iff $x_b == y_j, \exists y_j \in SHT[b]$ else ϕ (random)
   - Cloud server returns the results $R = \{r_1, ..., r_b, ..., r_\beta\}$ to client.
   - Client checks the count of correct secret values received from cloud server.

OUTPUT:

Client outputs $|S \cap R|$, where $S = \{s_1, ..., s_b, ..., s_\beta\}$ and $R = \{r_1, ..., r_b, ..., r_\beta\}$

Figure 5. DPSI-CA construction

figure 6. The protocol involves three parties - a client, cloud server and backend server. The protocol involves each user's device maintaining a count vector V representing the number of times a user visited a location. The vector V associates each index with a predetermined location of interest. Additive secret sharing is used to distribute shares of V to the servers. The servers then aggregate shares received from multiple users.

Security Discussion. From each server's view, it obtains a share of the count vector from each client. The share reveals nothing about the count vector, and hence, the server cannot learn an individual user's location trace or visits. The aggregate of shares received from multiple users are uniformly random. The recombination of aggregate shares results in the correct aggregate of count vectors, the intended result of the protocol, available to servers and clients. Neither the client nor the servers can deduce anything more than the histogram of hotspots, as the aggregate does not reveal the count vector of an individual or subset of clients.

PARAMETERS:

- Count vector V of size n
- A client $\mathcal{C}$, a backend server $\mathcal{S}$, and a cloud server $\mathcal{H}$
- Additive secret sharing scheme: If a is the original item, $[a]$ represents the set of shares. A two-out-of-two secret sharing scheme results in $[a] = \{a_1, a_2\}$ such that $a_1 + a_2 = a$

INPUTS:

- Client $\mathcal{C}$ has count vector V
- Backend server and Cloud server $\mathcal{H}$ have no input.

PROTOCOL:

1. Each user device maintains count vector V from location logs. If user visits location *loc* associated with index i in V, then increment $V[i]$
2. If user is positively diagnosed, obtain shares of V, $[V] = \{V_1, V_2\}$ such that $V_1[i] + V_2[i] = V[i], \forall i \in [1, \ldots, n]$. Send V_1 to backend server $\mathcal{S}$ and V_2 to cloud server $\mathcal{H}$
3. Each server, $\mathcal{S}$ and $\mathcal{H}$, maintains an aggregate of shares received, $V_{\mathcal{S}}$ and $V_{\mathcal{H}}$ respectively. $V_{\mathcal{S}} = V_{\mathcal{S}} + V_1$ and $V_{\mathcal{H}} = V_{\mathcal{H}} + V_2$
4. On aggregating a threshold number of shares, cloud server sends $V_{\mathcal{H}}$ to backend server $\mathcal{S}$. Backend server recovers and outputs hotspots histogram, the correct aggregate, by recombining the aggregate of shares as $V_{Hist} = V_{\mathcal{S}} + V_{\mathcal{H}}$

OUTPUT:
Hotspots histogram V_{Hist}

Figure 6. Hotspots histogram construction

Chapter 5

DISCUSSION AND ANALYSIS

The end-to-end contact tracing framework is implemented and tested. Section 5.1 describes relevant implementation considerations, algorithm choices, and parameter values for the protocols described in sections 4.2, 4.3 and 4.4. Section 5.2 describes the platform used for evaluation, the tests carried out, and analyses the performance.

5.1 Implementation Details

The three protocols, BT plus GPS based contact tracing, DPSI-CA protocol, and hotspots histogram computation, described in sections 4.2, 4.3 and 4.4, involve three parties - client app on mobile device, cloud server and backend server. The client functionality for the protocols is developed in Java as an Android mobile application. The backend and cloud servers are implemented in Java using the Spring framework.

5.1.1 BT plus GPS Contact Tracing Protocol

The BT plus GPS protocol extends and augments legacy decentralized BT based contact tracing approaches with location features. The implementation of proposed protocol builds on the legacy DP3T [7] Android app and server code available on Github at http://github.com/dp-3T/dp3t-sdk-backend and http://github.com/DP-3T/dp3t-sdk-android. The location module is inspired with ideas from the Safe Paths approach [13], whose code is available at https://github.com/

Path-Check/safeplaces-dct-app. The code for BT plus GPS protocol is available at http://github.com/ASU-FACT.

The client application involves the following major modules:

- Bluetooth server - to broadcast rotating proximity identifiers/tokens, register handshakes
- Bluetooth client - to scan for nearby devices, receive rotating proximity identifiers/tokens and store associated metadata like signal strength, duration of handshake
- Sync - to sync with backend server periodically to get exposure alerts, either through download of infected users tokens, or using Delegated Private Set Intersection - Cardinality protocol, as well as to upload tokens when positively diagnosed with the disease
- Cryptography - to help with key generation, rotation, psuedo random generation, encryption and hashing
- Database - to assist in create, read, update of delete of data in relevant tables, including tokens broadcast, tokens received, infected users tokens and associated metadata.
- Location - module to periodically log user's location and get associated geohashes

Similar to the Safe Paths approach [13], When a user is at a given set of coordinates, there is a radius r within which another user is said to be in close proximity. Points in the circle of radius r may lie in a neighboring geohash. Hence, for a given location, the geohash of the exact coordinates, as well as a set of neighboring geohashes covering the circle of proximity, is determined. This is done by considering the set of nearby points at a distance r along the cardinal and ordinal directions and determining the geohash of these points as well. The geohash of nearby points may overlap, and thus

there may be upto $N+1$ geohashes for a given location.

The broadcast bluetooth tokens are rotated every fifteen minutes. Location logs are recorded every five minutes. When storing hash of received token with geohash and timestamp, AES is chosen over SHA-256 as it is faster. The resulting hash has 128 bits. The app syncs with the backend server every two hours to receive tokens of infected users (legacy approach). The app can instead invoke DPSI-CA protocol to securely match tokens and receive exposure alerts. The app deletes tokens broadcast and received that are older than 14 days, which is the infectious period of COVID-19. The backend server exposes API endpoints, handling user requests to fetch and upload infected users' tokens. It maintains a database to store tokens, where tokens older than 14 days are deleted.

5.1.2 DPSI-CA Protocol

The implementation for DPSI-CA protocol is available at www.github.com/ASU-SC. In the DPSI-CA protocol, the client app only needs to upload the tokens received from other users in close proximity. Both cloud and backend servers are implemented in Java using Spring framework. The servers expose RESTful APIs to communicate and consume services. The Cloud server exposes a getMatches API, used by client device to provide its list of received tokens and to get count of matches/exposures in return. The backend server exposes a getPolynomials API used by the cloud server to provide the CHT hash functions and get the polynomial coefficients for each bin of the hash tables. The tokens are 128 bits long. To support polynomial interpolation and evaluation for such large data, the Java library implementations for polynomial interpolation using Lagrange's algorithm, and polynomial evaluation using Neville's

algorithm, are modified and extended to support the Java BigDecimal datatype. The cuckoo hashing implementation utilizes two hash functions to insert the user uploaded tokens into bins such that there is at the most one item per bin. The same two hash functions are used by the backend server to insert infected tokens into the simple hash table. AES is used as the hash function algorithm.

5.2 Performance

The performance of BT plus GPS based contact tracing when carried out using the legacy approach to determine matches by downloading infected tokens from the backend server is analysed. The major costs involve storage of tokens, upload and download of tokens and time taken for matching tokens to get exposure alerts. If a user generates a new token every 15 minutes and runs the proximity tracing process for approximately 18 hours a day, then each user sends 72 distinct, 128-bit tokens per day. Assuming that the user meets people and receives the same number of tokens, then each user device has a total of $n = 1008 \approx 1000$ tokens over a 14-day period. With 1,000 new cases per day, the backend server will receive $N \approx 1000 \times 1000 = 10^6$ new tokens per day.

Token storage: Storing both broadcast and received tokens for a 14 day period requires $\approx 31KB$ on client device. Assuming the server stores tokens for 15 days to accommodate for offline clients, the total storage needed is $\approx 0.25GB$ for 1000 daily new cases, and $\approx 1.25GB$ for 5000 daily new cases. If the client uses legacy approach to download new infected tokens uploaded for that day and match with received tokens on device, the client incurs download and storage costs.

Testing platform configuration: The client application is installed as a Java

Android app on a Google Pixel 3 device with Snapdragon 845 processor, 4 GB RAM and 64 GB storage. The backend server and cloud server are deployed on an AWS m5.2xlarge instance with 8 vCPUs, 32 GB memory and upto 10 Gbps network bandwidth.

Table 2 summarizes the time taken for the contact tracing framework, specifically token matching process, when using the BT plus GPS protocol without DPSI-CA protocol for token matching. The process involves the app on client device sending a request to fetch infected users' tokens from backend server and then matching with the received tokens stored on the device. The matching takes $O(n)$, where n is the number of tokens on device. The time complexity is linear as the server tokens are stored in a hashset and only membership test for each item in received tokens' list is required. The major cost of BT plus GPS based contact tracing without DPSI-CA involves bandwidth and storage, with 0.25 GB required if 1000 new cases are encountered daily.

Table 2. BT plus GPS Contact Tracing

#Client tokens(n)	#Server tokens(N)		
	500,000	750,000	1,000,000
1000	13.3	17.05	25.2
2000	13.4	17.06	25.3
3000	13.4	17.06	25.3

Time taken to match tokens in seconds (s)
n = Number of tokens on client device, N = Number of tokens on server

Table 3 lists the time it takes to interpolate for different number of tokens, each of size 128 bits. The interpolation is carried out by the backend server, and the results

reported are for the time taken on AWS m5.2xlarge instance with 8vCPUs and 32 GB memory.

Table 3. Time taken for polynomial interpolation in milliseconds (ms)

#Tokens	Time (ms)
5	0.048
10	0.137
50	6.118
100	49.27
250	871
500	8159

Table 4 summarizes the time taken by the backend server, deployed on AWS m5.2xlarge instance, to generate the polynomials for all bins, the major computation invovled in the DPSI-CA protocol. The code has been parallelized to run the polynomial interpolation on 7 threads. With 2 hash functions, the number of tokens in the hash table is doubled. The number of tokens per bin varies as per the hash function distribution. The polynomial interpolation for separate bins can be parallelized on more threads, resulting in α times speedup with α threads for parallel interpolation.

Table 4. Time taken by backend server to generate all polynomials in Seconds (ms)

# Bins (β)	# Server tokens (N)	Time (s)
40,000	500,000	19.2
80,000	500,000	14.2
80,000	1,000,000	37.7
100,000	1,000,000	34.7

The DPSI-CA protocol performance is compared with other works, including the Google Apple approach [5], DP3T [7], PACT [14], Epione [26] and Catalic [27], with respect to security and privacy guarantees, infrastructure requirements and client side cost in terms of computation and communication. The comparison is

presented in Tables 5 and 6. The method of evaluation followed is as explained in [27], and outlined briefly here. The Google Apple approach, DP3T and PACT publicly release tokens of diagnosed users, and hence they are all vulnerable to identification of diagnosed user. In the Google Apple approach, keys or seeds used to generate the tokens are publicly available, and hence allowing an adversary to learn the travel route of an infected user. Epione and Catalic keep the tokens private and hence secure against these vulnerabilities, similar to DPSI-CA protocol proposed in this work.

Table 5. DPSI-CA protocol comparison with other approaches

Protocols	Linkage Attack		System Req.	
	Travel Route	Infection Status	#Rounds	#Servers
Google Apple	yes	yes	1/2	1
DP3T	no	yes	1/2	1
PACT	no	yes	1/2	1
Epione	no	no	2	2
Catalic	no	no	1	3
DPSI-CA	no	no	1	2

Comparison of DPSI-CA with other contact tracing systems, in terms of privacy and infrastructure requirements. Travel route refers to learning the travel route of diagnosed user, while infection status refers to identification of diagnosed user. #Rounds is the number of interaction rounds between client and server.

Each user has $k = 144$ new tokens per day and receives a total of $n = 2^{11}$ tokens approximately over the 14 day infection window, according to the Google Apple approach. Also, with $K = 2^{15} = 32768$ new cases per day, $N = 2^{26}$ new tokens are added daily.

In the Google Apple approach, the client device downloads $14K$ keys per day. Each key is 128 bits long, resulting in 7.34 MB of communication cost. The device needs to

compute $14Kk = 66,060,288$ AES operations, taking 0.33 seconds to complete the contact tracing query on a phone with 1.99 GHz processor.

The DP3T approach utilizes a Cuckoo filter to share the tokens of diagnosed users. They store a 56-bit fingerprint with each item. With $N = 2^{26}$ new diagnosed tokens, the client incurs a communication cost is $2^{26} \times 56 = 469.76$ MB when downloading the Cuckoo filter. For computation, the device computes $2n$ AES hash functions, taking 0.02 milliseconds.

For the PACT approach, the client device downloads $2^{26} \times 128(\text{bits}) = 1073.74$ MB for $N = 2^{26}$ new diagnosis tokens. Its running time is considered negligible as it does not carry out any cryptographic operations.

In Epione, private set intersection using Private Information Retrieval is used, for which the client device incurs 1.79 MB and takes 394 milliseconds. For Catalic, with 1 backend and 2 cloud servers, each running with a single thread, the protocol requires 0.86 milliseconds 96 KB.

For the DPSI-CA protocol proposed using one cloud server and backend server, the client device has 1 round of interaction with the cloud server and backend server, and is required to download n results from the cloud server, equal to the number of tokens that it sends in the query. Thus the communication cost is $0.032MB$. The client device computes n AES hash functions to encrypt the tokens and generates $\beta = n$ secret values, where β is the number of bins, taking a total of 208 ms.

Table 6. DPSI-CA protocol client performance comparison with other approaches

Protocols	Client	
	Runtime (ms)	Comm. Cost (MB)
Google Apple	331.96	7.34
DP3T	0.02	469.76
PACT	neg	1073.74
Epione	394.01	1.27
Catalic	0.86	0.095
DPSI-CA	208	0.032

Comparison of DPSI-CA with other contact tracing systems, in terms of client runtime and communication costs. Travel route refers to learning the travel route of diagnosed user, while infection status refers to identification of diagnosed user. Each user has 2^{11} tokens. neg refers to negligible cost

Chapter 6

CONCLUSION AND FUTURE WORK

In this work, a secure, scalable and efficient Contact Tracing framework is proposed and evaluated. A hybrid, Bluetooth plus GPS based protocol for proximity detection is described, which is secure and robust against the vulnerabilities present in several existing Contact tracing applications already deployed. DPSI-CA, a novel protocol for private matching of tokens for exposure detection is proposed. DPSI-CA leverages server-aided computation to reduce computation cost on client devices. Moreover, a protocol to utilize location data collected on user devices without compromising individual user privacy so as to determine hotspots, areas of high-risk, is presented. The framework proposed can be improved with insights from further analysis and tests, including testing with multiple devices, different configurations and in scenarios reflecting real-world deployment. The DPSI-CA protocol can be made faster and more efficient by improving the polynomial interpolation. Further evaluation is required to determine optimal number of bins in a parallel processing environment. With further evaluation and testing, the proposed work can then be compared with existing work. Using Counting Bloom Filter and Perfect Hashing in place of polynomial interpolation will be explored. A more robust implementation of the hotspots histogram computation is required, followed by evaluation of utility. Techniques to avoid a predetermined of list of places of interest will be explored.

REFERENCES

[1] *Pneumonia of unknown cause – china*, 2020. [Online]. Available: https://www.who.int/csr/don/05-january-2020-pneumonia-of-unkown-cause-china/en/.

[2] *Coronavirus: China's first confirmed covid-19 case traced back to november 17*, 2020. [Online]. Available: https://www.scmp.com/news/china/society/article/3074991/coronavirus-chinas-first-confirmed-covid-19-case-traced-back.

[3] *Covid-19 coronavirus pandemic*, 2020. [Online]. Available: https://www.worldometers.info/coronavirus/.

[4] K. T. Eames and M. J. Keeling, "Contact tracing and disease control," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, vol. 270, no. 1533, pp. 2565–2571, 2003.

[5] *Apple and google partner on covid-19 contact tracing technology*, 2020. [Online]. Available: https://www.apple.com/newsroom/%202020/04/apple-and-google-partner-on-covid-19-contact-tracing-technology.

[6] "Covid watch," 2020. [Online]. Available: https://www.covid-watch.org.

[7] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, *et al.*, "Decentralized privacy-preserving proximity tracing," *arXiv preprint arXiv:2005.12273*, 2020.

[8] T. Coalition, "Temporary contact numbers protocol," *Retrieved September*, vol. 1, p. 2020, 2020. [Online]. Available: https://github.com/TCNCoalition/TCN.

[9] *Trace together*, 2020. [Online]. Available: https://www.tracetogether.gov.sg/.

[10] K. Pietrzak, "Delayed authentication: Preventing replay and relay attacks in private contact tracing.," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 418, 2020.

[11] H. Cho, D. Ippolito, and Y. W. Yu, "Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs," *arXiv preprint arXiv:2003.11511*, 2020.

[12] Y. Gvili, "Security analysis of the covid-19 contact tracing specifications by apple inc. and google inc.," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 428, 2020.

[13] A. Berke, M. Bakker, P. Vepakomma, R. Raskar, K. Larson, and A. Pentland, "Assessing disease exposure risk with location histories and protecting privacy:

A cryptographic approach in response to a global pandemic," *arXiv preprint arXiv:2003.14412*, 2020.

[14] J. Chan, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, S. Singanamalla, J. Sunshine, *et al.*, "Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing," *arXiv preprint arXiv:2004.03544*, 2020.

[15] S. Vaudenay, "Centralized or decentralized? the contact tracing dilemma," Tech. Rep., 2020.

[16] R. Raskar, I. Schunemann, R. Barbar, K. Vilcans, J. Gray, P. Vepakomma, S. Kapa, A. Nuzzo, R. Gupta, A. Berke, *et al.*, "Apps gone rogue: Maintaining personal privacy in an epidemic," *arXiv preprint arXiv:2003.08567*, 2020.

[17] *Path check*. [Online]. Available: https://www.pathcheck.org.

[18] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," 1966.

[19] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1243–1255.

[20] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?."

[21] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious prf with applications to private set intersection," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 818–829.

[22] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and N. Trieu, "Practical multiparty private set intersection from symmetric-key techniques," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1257–1272.

[23] B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Spot-light: Lightweight private set intersection from sparse ot extension," in *Annual International Cryptology Conference*, Springer, 2019, pp. 401–431.

[24] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 515–530.

[25] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, S. Saxena, K. Seth, M. Raykova, D. Shanahan, and M. Yung, "On deploying secure computing: Private intersection-sum-with-cardinality," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2020, pp. 370–389.

[26] N. Trieu, K. Shehata, P. Saxena, R. Shokri, and D. Song, "Epione: Lightweight contact tracing with strong privacy," *arXiv preprint arXiv:2004.13293*, 2020.

[27] T. Duong, D. H. Phan, and N. Trieu, "Catalic: Delegated psi cardinality with applications to contact tracing," in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2020, pp. 870–899.

[28] S. Dittmer, Y. Ishai, S. Lu, R. Ostrovsky, M. Elsabagh, N. Kiourtis, B. Schulte, and A. Stavrou, "Function secret sharing for psi-ca: With applications to private contact tracing," *arXiv preprint arXiv:2012.13053*, 2020.

[29] *Coronavirus disease (covid-19)*. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019.

[30] W. McKibbin and R. Fernando, "The global macroeconomic impacts of covid-19: Seven scenarios," *Asian Economic Papers*, pp. 1–55, 2020.

[31] G. Niemeyer, *Geohash (2008)*.

[32] S. H. Standard, "Fips pub 180-1," *National Institute of Standards and Technology*, vol. 17, p. 15, 1995.

[33] M. Dworkin, "Recommendation for block cipher modes of operation. methods and techniques," National Inst of Standards and Technology Gaithersburg MD Computer security Div, Tech. Rep., 2001.

[34] D. Demmler, P. Rindal, M. Rosulek, and N. Trieu, "Pir-psi: Scaling private contact discovery," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 4, pp. 159–178, 2018.