Privacy-Preserving COVID-19 Contact Tracing App: A Zero-Knowledge Proof Approach

Joseph K. Liu[1], Man Ho Au[2], Tsz Hon Yuen[2], Cong Zuo[1], Jiawei Wang[1], Amin Sakzad[1], Xiapu Luo[3], Li Li[1]

[1] Faculty of Information Technology, Monash University, Australia
{joseph.liu}@monash.edu
[2] The University of Hong Kong, Hong Kong
[3] The Hong Kong Polytechnic University, Hong Kong

Abstract. In this paper, we propose a privacy-preserving contact tracing app for COVID-19. The app allows users to be notified, if they have been a close contact with a confirmed patient. Our protocol is the most comprehensive and balanced privacy-preserving contact tracing solution to date.

Our protocol strikes a balance between security, privacy and scalability. In terms of privacy, it allows all users to hide his past location and contact history with respect to the Government. Yet, all users can check whether he had a close contact with a confirmed patient without learning the identity of the patient. We use a zero-knowledge protocol to ensure that user privacy is protected. In terms of security, no user can send fake message to the system to launch a false positive attack. We give a formal security model and give a security proof for our protocol. In terms of scalability, we have implemented our protocol into Android smartphone and our evaluation result shows its practicality.

1 Introduction

The COVID-19 pandemic opens up many opportunities for computer scientists to contribute in digital information technology social for good. One general approach taken by government officials and industry people is to (fully or partially) automate the extensive task of manual contact tracing. The main goal of contact tracing is to identify and quickly reach out to the potentially infected virus carrier individuals, so that they can (or ultimately forced to) take further actions. The actions include get tested, self-quarantined, and/or to watch their symptoms more closely. Although such an identification-investigation cycle would prevent further virus spread at a dramatic rate if implemented correctly and in a determined and forceful manner, it comes with limitations as people may simply do not know whom they contacted or if they even know the contactee, they do not know how to reach them or they do not want to. A fully automated contact tracing application integrated in smart devices including phones, tablets, or watches can tackle such deficiencies associated to traditional manual contact tracing.

Smart devices are capable of knowing where they are (approximately) using GPS (Wifi, respectively). And hence, there has been much research into design and implementation of automated contact tracing protocols and applications recently. Most of the well-known projects are Bluthooth-base ones [22,9]. Both above-mentioned approaches are working based on the fact that smart devices can learn and possibly share a customer's location coupled with a timestamp with others. In fact, in addition to a digital device whereabouts, the current protocol designs and applications might reveal some metadata about user's smart device model, contact details, etc. Sharing data through such applications to prevent the further spread of virus by itself is not bad, unless it harnesses someone's privacy.

Apple Inc. and Google Inc. initiated an interoperable privacy-preserving contact tracing application for their mobile users [2,3]. Decentralized Privacy-Preserving Proximity Tracing (DP-3T) light and unlinkable are proposed in [27]. These were endorsed by Pan-European Privacy-Preserving Proximity Tracing (PEPP-PT). Two almost similar PACTs; east coast [22] and west coast [9] were also introduced. The PACT east coast was evolved from [8] and other works.

[12] provided a security analysis of the Apple and Google privacy-preserving contact tracing application. Vaudenay [28] analyses DP-3T and provides relay and replay attacks. He also proposes an interactive scheme which prevent those attacks without compromising cool privacy aspects of DP-3T. A non-interactive countermeasure for relay and reply attacks to DP-3T and both PACTs is introduced in [19] via 'delayed authentication'. This is a novel message authentication code (MAC) in which the verification step is done in two phases. At one phase key is not required, while in the other one the message is not needed.

Besides the discussed attacks and elegant solutions, there are other issues in DP-3T, PACTs, and Apple Inc. and Google Inc. designs to be addressed. In this work, we take an alternative approach and present a (centeralized) solution to the problem of privacy-preserving contact tracing based on zero-knowledge proof. Our solution relies on Bluetooth-enabled smartphone as the tracing device, and hence the app installed as our proposed system. Before stating our contributions let us fix our privacy goals.

1. The Government and the Medical Doctor only know the personal details of the confirmed patients, but nothing about their close contacts or activities including the names or the location of the contacts happening.
2. A close contact of a confirmed patient will be notified (e.g. for a necessary self-isolation) by the app. But the probability of a user to guess correctly of "who is the confirmed patient" among a list of close contacts is not better than a wild guess. For example, suppose Bob has met Alice, Andy and Peter on the 1st of May. Later on when Alice is confirmed as a patient, Bob will be informed he has been a close contact of a confirmed patient on the 1st of May but he cannot know which one exactly is the patient though. The probability of guessing correctly is 1/3.
3. The privacy of a patient will not be provided after getting confirmed as positive, if he/she continues to make close contact with other people after

the confirmation. But anyone who has been the close contact of the patient *before* his/her confirmation will not know the patient's identity. We believe this is reasonable as any confirmed patient should *not* continue to make close contact (e.g. taking public transportation) with other people. Otherwise the case should be reported immediately to prevent further spreading of the virus.

Note that we do not *guarantee* the privacy of confirmed patients will be revoked or patients can be traced, as it is not the goal of our system. In fact, if a confirmed patient does not want to be traced, he/she can simply not install the app, or just put the smartphone at home while going outside. We regard this issue as out of scope of this paper. The point of this privacy goal only means our system will not provide privacy to confirmed patient, while the patient may get other means to obtain the untraceability in which we do not think is appropriate though.

Hence our contributions are as follows:

- We have introduced a privacy-preserving contact tracing app using zero-knowledge Σ-protocol. It can ensure users can record their close contacts in a privacy-preserving yet authenticated way, which prevents anyone from sending faked identification information.
- When a user has been confirmed positive, he proves in zero-knowledge to the medical doctor of all his previous close contacts. The medical doctor publishes some information to the public so that only the particular person will be known of being contacted with a confirmed patient. Yet the person (including the public) still has no idea on who the patient is, or no better than a wild guess.
- The medical doctor or the Government knows nothing about the patient's contacts, including the location (which will not be collected from the beginning), the name or any identification information. But the person will be convinced that he is a close contact of a patient.
- The security of our protocol also ensures that no one is able to send any faked isolation messages: if a user is not a confirmed person, he should not be able to let anyone else to be convinced that he is a close contact of a patient. On the other side, a confirmed patient is also not able to let anyone who is not a close contact of him, being convinced as a close contact.
- We prove the cryptographic security of our protocol based on various standard mathematical assumptions.
- We implement our protocol into Android smartphone and our evaluation result shows that it is practical to be used.

1.1 Related Work

A privacy-preserving profile-matching (PPPM) algorithm based on proximity-based mobile social networking (PMSN) is proposed in Zhang et al. [30]. This algorithm enables users match their profiles without revealing any information

about their profiles. FindU is a PPPM introduced in [14]. FindU only shares necessary and minimal information about the private attributes of the participating users.

Another line of related research is the privacy-preserving location sharing in mobile social network [20,29,15,17,23]. However, most solutions in this area assume that the social network platforms know the location of each user and they provide a privacy-preserving searching function based on location.

Some privacy-preserving location-based services uses k-anonymity, dummy locations, fully homomorphic encryption (FHE) or private information retrieval (PIR) to protect location or query privacy. Some of these schemes are not secure (e.g., attacks in [18,16]), and some are not very practical (e.g., FHE and PIR).

Governmental Solutions An alternative approach to PPPM is the one used by the Singapore Ministry of Health employing Bluetooth and Wifi sensing based on the idea of EPIC [1]. EPIC allows a set of observed devices to be compared to a device that belongs to an infected individual. Singapore's TraceTogether app has some privacy concerns raised by Asghar et al. [4] and Tang [24].

The Australian Government launched the COVIDSafe app [4] that stores the contact's identification information inside the smartphone in an encrypted format. When someone is diagnosed with COVID-19, the Government will ask them who they have been in contact with. If they have the COVIDSafe app and provide their permission, the encrypted contact information from the app will be uploaded to the Government, who will then use the contacts captured by the app to support their usual contact tracing and call people to let them know they may have been exposed. The Israeli authorities launched a contact-tracing app called the HaMagen [13] which is similar to the Australian version in that it stores users' data on their devices, and users who are subsequently diagnosed with coronavirus must decide whether to release their location data to authorities. Obvious the privacy of this approach is not very high and users need to put a high level of trust to the Government.

The Korean Government uses mobile phone location data to track the movements of people who later test positive [25]. Because the technology uses GPS location data, and phone companies in South Korea require all customers to provide their real names and national government registration numbers, it is effectively impossible to avoid being tracked if you own a smartphone. After some users subject to quarantine requirements reportedly flouted tracking systems by simply leaving their phones at home, authorities announced plans to ask repeat offenders to begin wearing tracking wristbands.

Hong Kong uses similar technology to enforce quarantine, with users required to wear a wristband with a unique QR code that pairs to their smartphone. Users download an app called StayHomeSafe [26], which uses geofencing technology to track their movements.

[4] http://tiny.cc/fljqmz

Academic Solutions Decentralized Privacy-Preserving Proximity Tracing (DP-3T) is proposed in [27] by using a cryptographic hash function, a pseudorandom number generator and a pseudorandom function. The security issues like relay and replay attacks, and also the related fixes [28,19] are discussed above.

The authors of [21] use multi-party computation (MPC) and oblivious random access memory (ORAM) and propose a contact tracing system with a central party (the health authority (HA)). CAUDHT (Contact tracing Application Using a Distributed Hash Table) [7] is a decentralized peer-to-peer generalization of [21]. A decentralised messaging system for infected persons and their contacts is constructed based on a distributed hash table. Blind signatures are also employed to ensure the authenticity of the messages. The major drawback of this approach is the use of expensive garbled circuits for MPC in the tracing phase.

1.2 Paper Organization

In the rest of the paper, it is organized as follow. We present the assumptions that we use in our system and the threat model we apply in Section 2. Cryptographic primitives that will be used in our system are presented in Section 3. The details of our proposed system is given in Section 4. The implementation and evaluation result are presented in Section 5 and 6 respectively. We also discuss some practical considerations in Section 7. Finally we conclude our paper in Section 8.

2 Assumptions and Threat Model

2.1 Entities and Assumptions

We first introduce the entities involved in our system here. They include:

- Bulletin Board $\mathcal{BB}$: Used for the publication of public information. Once data has been put into it, no one can erase. It can be instantiated by using a blockchain system.
- User: It can be referred as the Tracing App inside the smartphone. In the rest of this paper, we use Alice and Bob to denote two users as close contacting persons.
- Medical Doctor $\mathcal{D}$: Patients can be only confirmed positive when they are seeing a medical doctor, who is also affiliated with a hospital.
- Government $\mathcal{GV}$: It is responsible to do the registration of users (with their app). We also assume $\mathcal{GV}$ has its own secret key $SK_{\mathcal{GV}}$ and public key $PK_{\mathcal{GV}}$. $PK_{\mathcal{GV}}$ is known to the public.

We have to make the following reasonable assumptions:

1. No one is able to modify the app, but the owner can read all data generated, stored and communicated through the app.

2. The app and the smartphone is connected to the Internet.
3. There is Bluetooth connectivity for the smartphone.
4. Users will not reveal their social activities (e.g. posting a photo with someone to Facebook) to the public.
5. Users will not share the secret keys with other people.

2.2 Threat Model

We assume all adversaries are *Honest-but-Curious*. That is, they are following the defined algorithms but they are curious to learn more than the allowed information. Also in our threat model, we only include *cryptographic attack*, while other means of attacks such as network attack (e.g. DDoS), software attack (e.g. modifying the app or sending computer virus), physical attack (e.g. shut down or physically destroy the smartphone) etc. are considered out of scope of this paper.

Under these conditions, we define the following threat model to our system and illustrate them in Figure 1:

1. [**Tracebility Completeness**] All close contacts of a confirmed patient will be informed for the date of contact. All *Honest-but-Curious cryptographic* adversaries should not be able to prevent any close contact of a confirmed patient from being informed.
   In Figure 1, Bob and John (as close contacts of Alice) should be informed as a close contact of a patient.
2. [**False Positive (case 1)**] Anyone who are not confirmed patients cannot pretend to be a confirmed one and send out messages to their close contacts (e.g. ask them to be self-isolated) pretending they are confirmed patients.
   In Figure 1, Andy cannot send any "close contact message" to Ben and Bob. Peter also cannot send any "close contact message" to Bob.
3. [**False Positive (case 2)**] Anyone who are confirmed patients can only send out messages to their close contacts (e.g. ask them to be self-isolated) but not other people who are not the close contact.
   In Figure 1, Alice cannot send any "close contact message" to Ben, who is not her close contact.

   `NOTE:` Here we do not consider medical doctor as adversary in case (2) and case (3).
4. [**Patient Privacy**] Anyone, except the medical doctor and Government, should not be able to find out the identification details of a confirmed patient. The adversaries include all close contacts of the patient (the close contact was made before the patient getting confirmed) and the public. In the case of a close contact (the adversary is a close contact of a confirmed patient), this is not better than a wild guess.
   In Figure 1, Ben does not know any information about the confirmed patient. Bob, a close contact of Alice, who has also met Andy and Peter, can guess Alice as the confirmed patient correctly with probability 1/3. John, another close contact of Alice, who has only met Alice, will know Alice is the confirmed patient.

5. **[Contact Privacy]** Anyone, *including the medical doctor and Government*, should not be table to find out the identification details of the close contact of a confirmed patient, including the location of the contact. The *cryptographic* adversary here tries to find out all details of the close contact of a confirmed patient.
   In Figure 1, no one except the owner of the app, knows any information about the close contact. For example, Andy does not know whom Alice has met, and Alice does not know whom Andy has met.

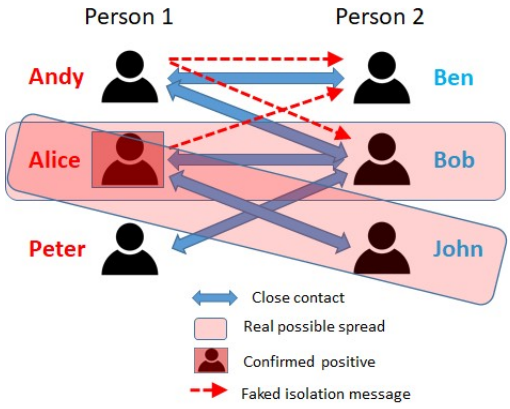

Fig. 1. Threat Model

3 Cryptographic Primitives

3.1 Signature Scheme

A signature scheme consists of three algorithms, which are defined as follow:

- $(SK, PK) \leftarrow \mathsf{KeyGen}(\lambda)$ as a PPT algorithm which, on input a security parameter $\lambda \in \mathbb{N}$, outputs a secret/public key pair (SK, PK).
- $\sigma \leftarrow \mathsf{Sign}(SK, M)$ which, on input a secret key SK and a message M produces a signature σ.
- $\mathsf{accept/reject} \leftarrow \mathsf{Verify}(PK, \sigma, M)$ which, on input a public key PK, a message M and a signature σ, returns accept or reject. If accept, the message-signature pair is *valid*.

A secure signature scheme should provide existential unforgeability against adaptive chosen-message attacks, in which we follow the standard definition given in [11]. A weaker notion, called *existential unforgeability under a weak chosen message attack (a.k.a. weakly unforgeable)* is defined in [6] such that the adversary submits all signature queries before seeing the public key.

3.2 Group Signature Scheme

A group signature allows a user to sign on behalf of the group, while the verifier only knows the signer is one of the users of the group without knowing the actual identity. There is a group manager who is responsible to setup the group (to publish the group public key) and to issue user secret key to each user in the group. He may also has the ability to open the signature, that is, to find out who the actual signer is.

There are several algorithms in a group signature scheme. For simplicity, we only state the algorithms related to our system here:

- $\sigma \leftarrow \mathsf{GSign}(USK, GPK, M)$ which, on input a user secret key USK (issued by the group manager), group public key GPK (generated by the group manager) and a message M produces a signature σ.
- $\mathsf{accept/reject} \leftarrow \mathsf{GVerify}(GPK, \sigma, M)$ which, on input a group public key GPK, a message M and a signature σ, returns accept or reject. If accept, the message-signature pair is *valid*.

We follow the standard security definition of group signature in [5], including anonymity and traceability which also implies unforgeability.

3.3 Mathematical Assumptions

Bilinear Pairings. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order q. u be a generator of $\mathbb{G}_1$ and g be a generator of $\mathbb{G}_2$. A function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map if the following properties hold:

- Bilinearity: $e(A^x, B^y) = e(A, B)^{xy}$ for all $A \in \mathbb{G}_1$, $B \in \mathbb{G}_2$ and $x, y \in \mathbb{Z}_q$;
- Non-degeneracy: $e(u, g) \neq 1$, where 1 is the identity of $\mathbb{G}_T$;
- Efficient computability: there exists an algorithm that can efficiently compute $e(A, B)$ for all $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$.

We give the assumption that we used for contact privacy, which is similar to the truncated decision $(q' + 2)$-ABDHE problem [10].

Definition 1 (Assumption 1). *Suppose that* $\tilde{u} \in_R \mathbb{G}_1$, $g \in_R \mathbb{G}_2$, $b \in_R \mathbb{Z}_q$, $Z_0 \in_R \mathbb{G}_T$ *and* $Z_1 = e(\tilde{u}, g)^{b^{q'+2}}$. *When given* $(\tilde{u}, \tilde{u}^b, \ldots, \tilde{u}^{b^{q'}}) \in \mathbb{G}_1$ *and* $g, g^b \in \mathbb{G}_2$, *no PPT adversary can distinguish* Z_0 *and* Z_1 *with non-negligible probability.*

3.4 Zero-Knowledge Proof

A zero-knowledge proof is a two-party protocol which allows one party to convince the other something is true without revealing anything else. In this paper, we are interested in zero-knowledge proof for NP language. Specifically, let R be a polynomial time decidable binary relation and L_R be the NP language defined by R, i.e., $L_R = \{x | \exists w \;\; s.t. (x, w) \in R\}$. We say w is a witness for statement x.

The zero-knowledge proof protocol we considered in this paper is known as Σ-Protocol, which is a 3-move protocol between prover P and verifier V such that the second message (from V to P) is just the random coins of V. A Σ-protocol between P and V satisfies the following properties.

- *Completeness*: If $x \in L_R$, prover P with auxiliary input w convinces V with overwhelming probability.
- *Special Soundness*: Given two transcripts (t, c, z) and (t, c', z') for statement x, there exists an algorithm which outputs w s.t. $(x, w) \in R$.
- *Honest Verifier Zero-Knowledge (HVZK)*: Given x and c, there exists an algorithm which outputs (t, z) such that (t, c, z) is indistinguishable to the real transcript between P with auxiliary input w and V.

We remark that Σ-Protocol can be converted to full zero-knowledge in the common reference string model using standard techniques. Also, it can be converted to non-interactive zero-knowledge argument in the random oracle model by replacing the random coins of the verifier with the output of a cryptographic hash function on the first message of the prover.

4 Our Proposed System

4.1 Overview

There are 4 phases in our system. In the **Registration Phase**, each user chooses his/her secret key and public key, and upload the public key to the Government website for registration. The Government hence can link the public key with the user's name or identity. This is to prevent a double-registration of each user. This process will repeat each day with a new key pair registered. A medical doctor gets additional a user secret key issued by his hospital affiliated, which is used to generate a group signature on behalf of the hospital.

In the **Meeting Phase**, each user's app will use Bluetooth to broadcast a package to other users' smartphones (and the apps) at a regular time interval (e.g. 1 minute). Upon receiving a threshold number of the same package within a certain timeframe (e.g. 15 minutes), the app will confirm the opposite user as a close contact. After a mutual validation (of package) process, the two apps will jointly generate two different credentials to be stored on each smartphone. The credential will be used later to prove to the doctor (in zero-knowledge) that the other person is a close contact, if the owner of the app is a confirmed patient.

In the **Medical Treatment Phase**, the patient is executing a zero-knowledge proof protocol with the medical doctor, to prove he/she has close contact with other people. Yet the doctor does not know anything regarding the identities, public keys or location of the close contact. The doctor signs the zero-knowledge proof using the group signature user secret key (on behalf of his affiliated hospital) and posts the signature together with the proof to the bulletin board for public checking.

In the **Tracing Phase**, each user check whether the new entry in the bulletin board is referring to themselves, by some computations using their own secret key.

4.2 Detail Description

We describe our proposed system in details here.

Setup Phase: In this phase, $\mathcal{GV}$ first generates the parameters and users register themselves to $\mathcal{GV}$. Users need to update the key everyday with $\mathcal{GV}$ until they have been confirmed positive. Details are outlined below:

1. **(Parameter Generation)** On input 1^λ, where $\lambda \in \mathbb{N}$ is a security parameter, let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order q such that q is a λ-bit prime, and let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map. $\mathcal{GV}$ selects generators $u, u_1, u_2 \in \mathbb{G}_1$ and $g, g_1, g_2 \in \mathbb{G}_2$. Let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a cryptographic hash function. $\mathcal{GV}$ also selects its secret and public key pair $(SK_G, PK_G) \leftarrow \mathsf{KeyGen}(\lambda)$. $\mathcal{GV}$ publishes the public parameters $(\lambda, H, PK_G, u, u_1, u_2, g, g_1, g_2)$. In practice, these can be also embedded into the user's app which can be downloaded from the official app stores.
2. **(User Registration)** Each day each non-confirmed user (e.g. Alice) chooses a secret key SK_A as $a \in \mathbb{Z}_q$ and computes the public key PK_A as $A = g^a$. She registers herself to $\mathcal{GV}$ by uploading her personal information and PK_A [5]. $\mathcal{GV}$ also randomly generates an identifier $\mathtt{ID}_A \in \mathbb{Z}_q$ for Alice. $\mathcal{GV}$ generates a signature $\sigma_A \leftarrow \mathsf{Sign}(SK_G, \{\text{``-VE''}, PK_A, \mathtt{ID}_A, \mathtt{DATE}\})$ and sends $\sigma_A$ and $\mathtt{ID}_A$ back to Alice, where $\mathtt{DATE}$ is the current date. Alice checks the signature by running $\mathtt{Verify}(PK_G, \sigma_A, \{\text{``-VE''}, PK_A, \mathtt{ID}_A, \mathtt{DATE}\})$. If it is valid, Alice stores σ_A and $\mathtt{ID}_A$ in her app. Otherwise, she aborts.
   NOTE: $\mathcal{GV}$ will give a signature with $\{\text{``+VE''}, PK_A, \mathtt{ID}_A, \mathtt{DATE}\}$ (instead of the normal ``-VE'' message) to any user who has been confirmed positive. This is to distinguish a confirmed case from others. The public key of confirmed users will not be updated as well.
3. **(Doctor's Extra Step)** Each medical doctor $\mathcal{D}$ gets additional the group signature user secret key GSK from the hospital manager (who acts as the group manager of the group signature) in the app. Each hospital also publishes the group signature group public key GPK.

Meeting Phase: In this phase, each non-confirmed user (e.g. Alice) will use bluetooth to broadcast the hash $h_A = H(\text{``-VE''}, \mathtt{ID}_A, PK_A, \sigma_A)$ to the surrounding people periodically (e.g. every minute). For any confirmed user, a ``+VE'' package (e.g. *without hashing* $(\text{``+VE''}, \mathtt{ID}_P, PK_P, \sigma_P)$ denoting the owner of the app who has been confirmed by the medical doctor as positive) will be broadcasted instead. If it has been received, other users should report to $\mathcal{GV}$ immediately after verifying the signature σ_P. Otherwise once another user (e.g. Bob) has received a number of the same hash broadcast within a certain time (e.g. receive 15 packages in 15 minutes), they (Alice and Bob) are considered as close contact. In below, we describe a protocol executed between Alice and Bob so that Alice will record Bob's information as her close contact. Bob will also Alice's information as his close contact at the end of the protocol.

[5] $\mathcal{GV}$ may record the related identification information (e.g. name, phone, email) of the user upon the registration. (If it is not the first day, the recording of identification information part is not needed.)

1. **(Package Validation)** After receiving (a threshold number of) Alice's hash h_A, Bob('s smartphone) pairs with Alice('s smartphone) and Bob needs to validate Alice's package. Bob first asks Alice to send him the tuples $(\texttt{ID}_A, PK_A, \sigma_A)$. Bob computes $h'_A = H(\texttt{''-VE''}, \texttt{ID}_A, PK_A, \sigma_A)$ and checks if $h_A = h'_A$. Bob aborts if it is not equal. Otherwise Bob continues and randomly generates a challenge number $r_B \in_R \mathbb{Z}$ and sends r_B to Alice. Alice uses her SK_A ($a \in \mathbb{Z}_q$) to generate a Schnorr signature on the message r_B, as follow:

   (a) Randomly choose $k \in_R \mathbb{Z}_q$.
   (b) Compute $t = H(g^k, r_B)$.
   (c) Compute $s = k - at \bmod q$.
   (d) Output the signature $\sigma'_A = (s, t)$ for the message r_B.

   Alice sends σ'_A to Bob for verification. Bob first verify PK_A ($A \in \mathbb{G}_2$) by running

   $$\texttt{Verify}(PK_G, \sigma_A, \{PK_A, \texttt{ID}_A, \texttt{DATE}\}). \tag{1}$$

   If it is valid, Bob verifies Alice's Schnorr's signature $\sigma'_A = (s, t)$ by checking if

   $$t = H(g^s A^t, r_B)$$

   If it is equal, Bob stores Alice's package $(\texttt{ID}_A, A, \sigma_A)$ in his app. Otherwise, aborts the protocol.

   On the opposite side, Bob does the same to let Alice validate his package. If the validation is valid, Alice also stores Bob's package $(\texttt{ID}_B, B, \sigma_B)$ in her app.
2. **(Identity Mutual Commitment)** Alice and Bob need to store each other's identification information and later on generate a zero-knowledge proof to $\mathcal{D}$ as a close contact to a patient (if either of them is confirmed). In order to let the proof generated correctly, we need to have an additional mutual commitment in this phase.

   Bob uses his secret key $b \in \mathbb{Z}_q$ and Alice's identifier $\texttt{ID}_A$ to generate

   $$\sigma''_B = u^{\frac{1}{H(\texttt{ID}_A)+b}}$$

   and sends σ''_B to Alice. Alice checks if

   $$e(\sigma''_B, g^{H(\texttt{ID}_A)} B) = e(u, g)$$

   If it is equal, Alice stores $(B, \texttt{ID}_B, \sigma''_B, \texttt{DATE})$ in her app.

   On the opposite side, Alice uses her secret key $a \in \mathbb{Z}_q$ and Bob's identifier $\texttt{ID}_A \in \mathbb{Z}_q$ to generate

   $$\sigma''_A = u^{\frac{1}{H(\texttt{ID}_B)+a}}$$

   and sends σ''_A to Bob. Bob checks if

   $$e(\sigma''_A, g^{H(\texttt{ID}_B)} A) = e(u, g)$$

   If it is equal, Bob stores $(A, \texttt{ID}_A, \sigma''_A, \texttt{DATE})$ in his app.

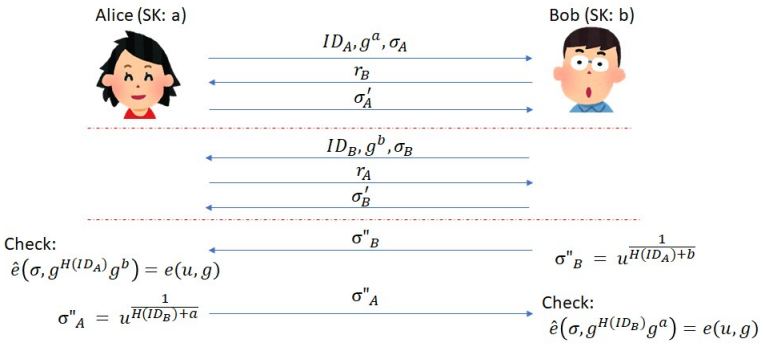

Fig. 2. Meeting Phase

The Meeting Phase is illustrated in Figure 2.

Medical Treatment Phase: In this phase, the Alice has been confirmed as positive in the hospital. Alice tells the medical doctor $\mathcal{D}$ about all her close contact during the possible spreading days. In order to provide privacy, Alice will not directly give $\mathcal{D}$ her close contacts' identifiers or public keys. Instead, she will generate a pseudo-public key of each of her close contact (e.g. Bob) together with a zero-knowledge proof from the mutual commitment generated in the **Meeting Phase** Step (2) to prove that she has contacted with the people . $\mathcal{D}$ then publishes the pseudo-public key to $\mathcal{BB}$ and the public can check whether this pseudo-public key is related to themselves later.

$\mathcal{D}$ and Alice execute the following protocol:

1. **(Authentication of Alice)** $\mathcal{D}$ first authenticates Alice by executing **Meeting Phase** Step (1) (Package Validation) and obtain her identifier $\texttt{ID}_A$.
2. **(Contact Retrieval)** For each possible spreading date in the past $\texttt{DATE}_i$ (e.g. Suppose today is `14th May`. The possible spreading dates will be `1st May`, `2nd May`, ..., `14th May`.) Alice retrieves her close contacts. Suppose Alice has met Bob at 13th May, Alice retrieves $(\texttt{ID}_B, B, \sigma''_B, \texttt{13th May})$ from her app.
3. **(Pseudo-Public Key)** Alice generates the pseudo-public key for Bob, by first randomly chooses $x \in_R \mathbb{Z}_q$ and computes:

$$h = e(u, g)^x, \qquad \widehat{B} = e(u, B)^x = e(u, g^b)^x = h^b \tag{2}$$

   and sends $(h, \widehat{B})$ to $\mathcal{D}$.
4. **(Zero-Knowledge Proof)** Alice needs to prove to $\mathcal{D}$ that $(h, \widehat{B})$ is correctly form. *Correct* means Alice has received a valid signature σ''_B under the public key $B = g^b$ and $\widehat{B} = h^b, h = e(u, g)^x$. Note that $\mathcal{D}$ also knows Alice's identifier $\texttt{ID}_A$. Conceptually, Alice needs to prove in zero-knowledge that

$$PK\{(\sigma''_B, B, x) : h = e(u, g)^x, \widehat{B} = e(u, B)^x, e(\sigma''_B, g^{H(\texttt{ID}_A)} B) = e(u, g)\}. \tag{3}$$

In order to instantiate this proof, Alice first randomly generates $s_1, s_2, t \in_R \mathbb{Z}_q$ and computes

$$A_1 = g_1^{s_1} g_2^{s_2}, \qquad A_2 = B g_1^{s_2}, \qquad C = \sigma_B'' u_1^t$$

Alice sends A_1, A_2, C to $\mathcal{D}$ and proves that

$$PK\{(s_1, s_2, t, \alpha_1, \alpha_2, \beta_1, \beta_2, x) : A_1 = g_1^{s_1} g_2^{s_2} \wedge A_1^x = g_1^{\alpha_1} g_2^{\alpha_2} \wedge$$

$$A_1^t = g_1^{\beta_1} g_2^{\beta_2} \wedge h = e(u,g)^x \wedge \widehat{B} = e(u, A_2^x g_1^{-\alpha_2}) \wedge$$
$$e(C u_1^{-t}, g^{H(\mathtt{ID}_A)} A_2 g_1^{-s_2}) = e(u,g)\} \tag{4}$$

This can be turned into a non-interactive zero-knowledge proof, by the following algorithm:

[Proof Generation]
(a) Randomly choose $r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8 \in_R \mathbb{Z}_q$.
(b) Compute

$$T_1 = g_1^{r_1} g_2^{r_2}, \qquad T_2 = A_1^{-r_6} g_1^{r_4} g_2^{r_5}, \qquad T_3 = A_1^{-r_3} g_1^{r_7} g_2^{r_8},$$

$$T_4 = e(u,g)^{r_6}, \quad T_5 = e(u, A_2^{r_6} g_1^{-r_5}), \quad T_6 = e(u_1^{r_3}, g^{H(\mathtt{ID}_A)} A_2) e(C^{r_2} u_1^{-r_8}, g_1)$$

(c) Compute the hash

$$c = H(T_1, T_2, T_3, T_4, T_5, T_6, \widehat{B}, h, A_1, A_2, C, \mathtt{DATE})$$

where `DATE` is the current date.
(d) Compute

$$\begin{aligned}
z_1 &= r_1 - c s_1 \bmod q \\
z_2 &= r_2 - c s_2 \bmod q \\
z_3 &= r_3 - c t \bmod q \\
z_4 &= r_4 - c s_1 x \bmod q \\
z_5 &= r_5 - c s_2 x \bmod q \\
z_6 &= r_6 - c x \bmod q \\
z_7 &= r_7 - c s_1 t \bmod q \\
z_8 &= r_8 - c s_2 t \bmod q
\end{aligned}$$

(e) Output the proof $\pi : (c, z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8)$.

[Proof Verification] Compute

$$\begin{aligned}
T_1' &= A_1^c g_1^{z_1} g_2^{z_2} \\
T_2' &= A_1^{-z_6} g_1^{z_4} g_2^{z_5} \\
T_3' &= A_1^{-z_3} g_1^{z_7} g_2^{z_8} \\
T_4' &= h^c e(u,g)^{z_6} \\
T_5' &= \widehat{B}^c e(u, A_2^{z_6} g_1^{-z_5}) \\
T_6' &= \left(\frac{e(C, g^{H(\mathtt{ID}_A)} A_2)}{e(u,g)} \right)^c e(u_1^{z_3}, g^{H(\mathtt{ID}_A)} A_2) e(C^{z_2} u_1^{-z_8}, g_1)
\end{aligned}$$

Accept the proof if and only if

$$c = H(T_1', T_2', T_3', T_4', T_5', T_6', \widehat{B}, h, A_1, A_2, C, \texttt{DATE})$$

5. **(Publish Pseudo-Public Key)** If the proof is correct, $\mathcal{D}$ generates a group signature $\sigma_D \leftarrow \mathsf{GSign}(USK, GPK, M)$ on the message $M = (h, \widehat{B}, \texttt{DATE})$ and publishes $(\sigma_D, h, \widehat{B}, \texttt{DATE})$ into $\mathcal{BB}$. $\mathcal{D}$ also informs $\mathcal{GV}$ that Alice (with identifier $\texttt{ID}_A$ and public key A) has been confirmed as positive. $\mathcal{GV}$ will update its entry on Alice: $\{\texttt{``+VE''}, PK_A, \texttt{ID}_A, \texttt{DATE}\}$ and sign this entry every date (update DATE only) until Alice has been fully recovered.

Tracing Phase: By the end of each day (e.g. 23:59), each non-infected (or not tested positive) user executes the following step:

Bob scans through $\mathcal{BB}$ for all new entries. For each entry

$$(\sigma_D, h, \widehat{B}, \texttt{DATE})$$

Bob first retrieves his secret key SK_B ($b \in \mathbb{Z}_q$) corresponding to that DATE and checks if $\widehat{B} = h^b$. If yes, Bob then verifies the signature by running $\texttt{GVerify}(\sigma_D, GPK, \{h, \widehat{B}, \texttt{DATE}\})$. If it is valid, he has been in close contact with a confirmed patient at DATE.

4.3 Security Discussion

We now discuss the security of our system, according to the threat model we defined in Section 2.2.

1. [**Traceability Completeness**]

   Lemma 1. *Our system provides Traceability Completeness if our protocol is correct.*

   We only consider Honest-but-Curious cryptographic adversaries here. Therefore we consider Alice (the confirmed patient), the medical doctor $\mathcal{D}$, Government $\mathcal{GV}$ and the bulletin board $\mathcal{BB}$ are all *Honest-but-Curious*. That is, they all follow the defined algorithms to execute the computation. The remaining thing that can prevent Bob (the close contact of the confirmed patient) from being informed is the *correctness* of the post (by $\mathcal{D}$) on $\mathcal{BB}$. The correctness of the tuple $(\widehat{B}, h)$ posted on $\mathcal{BB}$, in which Bob will use it to identify himself as the close contact, can be easily seen from equation (2).
2. [**False Positive (case 1)**]

   Lemma 2. *Our system does not have Case 1 False Positive if the underlying group signature scheme (Section 3.2) is unforgeable.*

   In this case the adversary is a non-confirmed user (e.g. Andy in Figure 1) who wants to pretend a confirmed patient and convince his close contacts (e.g. Ben in Figure 1) that they are close contacts of a confirmed patient. Observe

that a user believes he is a close contact of a confirmed patient, only if they download the entries $(\sigma_D, h, \widehat{B}, \texttt{DATE})$ from $\mathcal{BB}$ and check: (1) σ_D is a valid group signature; and (2) $\widehat{B} = h^b$. The adversary who was a close contact of other user can compute $(h, \widehat{B})$ as in equation (2) (as the adversary knows the public key of the close contact). However, if the adversary is successful, the adversary should be able to forge the group signature σ_D which contradicts to the unforgeability of the underlying group signature scheme.

3. **[False Positive (case 2)]**

   Lemma 3. *Our system does not have Case 2 False Positive if the zero-knowledge proof in equation (3) is sound and the Boneh-Boyen signature [6] is weakly unforgeable.*

   In this case, the adversary (Alice) is a confirmed user who wants to convince the Doctor that a certain user, say, Ben, is a close contact (but in fact Ben is actually not). Let B' be the public key of Ben. Alice needs to produce a valid proof for equation (3). If the proof is sound, there exists an extractor[6] to extract witnesses (σ''_B, B, x) such that σ''_B is a Boneh-Boyen signature on ID_A under public key B such that $\hat{B} = e(u, B)^x$ and $h = e(u, g)^x$ for some x . For Ben to think he has been in close contact, $\hat{B}$ has to satisfy the relation that $DL_g(B') = DL_h(\hat{B})$. By the soundness of the proof, this requires $B' = B$. If Alice has never been in close contact with Ben, it means Alice has forged a Boneh-Boyen signature from Ben on $\texttt{ID}_A$. This is impossible if we assume Boneh-Boyen signature is unforgeable against weak chosen message attack[7]. It remains to argue that soundness for the proof of equation (4) indeed implies that of equation (3). The argument goes as follows. If the proof for equation (4) is sound, one could extracting witness $(s_1, s_2, t, \alpha_1, \alpha_2, \beta_1, \beta_2, x)$. Further, since $A_1 = g_1^{s_1} g_2^{s_2}$ and $A_1^x = g_1^{\alpha_1} g_2^{\alpha_2}$, we have $\alpha_2 = s_2 x$. We have

$$e(Cu_1^{-t}, g^{H(\texttt{ID}_A)} A_2 g_1^{-s_2}) = e(u, g).$$

   In other words, Cu_1^{-t} is a valid Boneh-Boyen signature on $\texttt{ID}_A$ under public key $(A_2 g_1^{-s_2})$. By $\hat{B} = e(u, A_2^x g_1^{-\alpha_2})$, we have $\hat{B} = e(u, A_2 g_1^{-s_2})^x$. Furthermore, we have $h = e(u, g)^x$. One could output witness $(Cu_1^{-t}, A_2 g_1^{-s_2}, x)$ as the witness of the proof for equation (3).

4. **[Patient Privacy]**

   Lemma 4. *Our system provides Patient Privacy from the public unconditionally, and from the patient's close contacts if the underlying signature scheme (Section 3.1) is unforgeable.*

   We consider two types of adversaries here. The first type is the general public, who also has the knowledge of what has been posted in $\mathcal{BB}$. The

[6] This requires treating H as random oracle

[7] Note that we assume $\texttt{ID}_A$ is chosen by the government or is the hash of some seed value chosen by Alice. This assumption restricts the message the attacker could obtain and allows us to reduce to the weak chosen message security of the Boneh-Boyen signature.

second type is the close contact of a confirmed patient, who want to guess which particular close contact is the confirmed patient. Note that we do not consider $\mathcal{D}$ or $\mathcal{GV}$ as the adversary here, as they are supposed to know the identity of the patient.

For the first type, we consider the information posted in $\mathcal{BB}$: $(\sigma_D, h, \widehat{B}, \texttt{DATE})$. $h = e(u,g)^x$ is generated by *Alice* (the confirmed patient) where x is a random number (and u, g are the public generators). $\widehat{B} = e(u,B)^x$ where B is the public key of the patient's close contact. It is obvious that the tuple $(h, \widehat{B})$ does not contain any information about the patient Alice. σ_D is a group signature on the message $(h, \widehat{B}, \texttt{DATE})$ which also does not have any information related to the patient (but just the hospital name where the patient went to).

For the second type, we can use a reduction proof that we use the adversary (Bob here, who is able to distinguish who the patient is) to output a forged signature. Suppose Bob has met n close contacts at DATE. He is also confirmed to be the close contact for a patient for that date DATE. Suppose the probability of guessing correctly of the patient among his contact is ρ. If we have $\rho > 1/n + \epsilon$ where ϵ is the negligible probability, the adversary should be able to distinguish the patient from the other users with non-negligible probability. Observe that Bob has collected all transactions between all users, and he can also see the tuple $(\sigma_D, h, \widehat{B}, \texttt{DATE})$ posted in $\mathcal{BB}$. In order to distinguish the patient and the others, Bob has to make use of the tuple in $\mathcal{BB}$. σ_D is just the group signature from $\mathcal{D}$ on the message $(h, \widehat{B}, \texttt{DATE})$. $h = e(u,g)^x$ is randomly generated by the patient (from a random number x chosen by the patient), which does not give any information to distinguish. The only element that can be used is $\widehat{B} = e(u,B)^x$, where B is the public key given by Bob to his close contact in Step (1) of the **Meeting Phase**. To distinguish each close contact, Bob has to give different B to different close contact. In order to let the protocol to complete, the verification algorithm in equation (1) should go through for each contact. For $n > 1$, suppose $n = 2$. Bob has two valid signatures σ_B on B and $\sigma_{B'}$ on B'. But he can only get one signature from $\mathcal{GV}$ everyday. Bob then outputs the other signature as the forged one.

5. [**Contact Privacy**]

   Lemma 5. *Our system provides Patient Privacy if the zero-knowledge proof in equation (3) and its instantiations are zero-knowledge and the assumption 1 holds in the random oracle model.*

   If there exists an attacker $\mathcal{A}$ that can break the contact privacy, we can build an algorithm $\mathcal{B}$ to break assumption 1. $\mathcal{B}$ picks a random degree q' polynomial $F(b)$ and sets $u = \tilde{u}^{F(b)}$. $\mathcal{B}$ sets the public key of Bob as $B = g^b$. By using the random oracle model, $\mathcal{B}$ can simulate q' signatures of Bob $\sigma''_B = \tilde{u}^{\frac{F(b)}{b+H(\texttt{ID})}}$ by setting $-H(\texttt{ID})$ as the roots of $F(b)$.
   The information from the confirmed patient Alice about her close contact Bob are zero-knowledge proof for equation (3) and the pair $(\hat{B}, h)$. If there exists an attacker who can identify Bob, one could use standard game-hopping

technique to change the tuple from $(\hat{B}, h)$ to $(\hat{R}, h)$, where $\hat{R}$ is a random group element. In this case, $\mathcal{B}$ sets $h = e(\tilde{u}^{F(b)}, g^{ab})$ for some random $a \in \mathbb{Z}_q$. It implicitly sets $x = ab$. $\mathcal{B}$ sets $\hat{R} = e(\tilde{u}^{F(b)}, g^b)^x$ by using Z_0 or Z_1 as the term with degree $b^{q'+2}$. The difference between the settings $(\hat{B}, h)$ and $(\hat{R}, h)$ will be bounded by the advantage of breaking the assumption 1. The switch further requires that the zero-knowledge proof for equation (3) is simulation-sound because now the zero-knowledge simulator is using a fake tuple $(\hat{R}, h)$. Luckily, the Σ-Protocol we use is indeed simulation-sound. If the proof for equation (3) is zero-knowledge, then no one will be able to learn any information about Bob.

It remains to argue that actual instantiation, proof for equation (4), is indeed HVZK. It is easy to see that the three moves protocol itself is HVZK. The argument that auxiliary values, A_1, A_2 and C, leaks no information goes as follows. For any possible witness (B, σ), there exists a unique randomness (s_2, t) such that $A_2 = Bg_1^{s_2}$ and $C = \sigma''_B u_1^t$. Now, for each possible s_2, there is a unique randomness s_1 such that $A_1 = g_1^{s_1} g_2^{s_2}$. Thus, the simulator can pick random elements A_1, A_2, C and use the zero-knowledge simulator to simulate proof for equation (4) since these random elements will also be correctly formed. In other words, they also leak no additional information.

5 Implementation

We have implemented the contact tracing app on Android 8.0, and tested it on Android 8.0 and Android 10.0. Figure 3 illustrates its architecture. The contact tracing app consists of four main modules, namely Bluetooth service, device discovery service, Crypto manager, and utility module. When the app is installed for the first time, the Crypto manager will be initialised with a set of operations such as key generation, downloading public keys or parameters from other parties, etc. Then, the app will start two background services which will interact with other components.

• Crypto manager module encapsulates most of aforementioned operations of Crypto algorithms, such as signing, verification, and etc. The module also helps the user generate key pairs($(SK_A$,$PK_A)$) on a daily basis. Then, it will upload the user's personal information and PK_A to $\mathcal{GV}$ for finishing the registration, and waits for the identifier $\texttt{ID}_A$ and the signature σ_A from $\mathcal{GV}$, which will be stored in the storage module along with SK_A, PK_A and other user information.

•The device discovering service is a background service listening to nearby Bluetooth advertising packets and filtering out irrelevant packets. When the number of packets received from other devices running our contact tracing app exceeds a predefined threshold, which is set to 15 packets within 15 minutes from the same sender by default, it will pair with that device for further communication.

•The Bluetooth service is another background service listening to nearby Bluetooth pair and connection requests. It also advertise the user's own hash of message periodically to nearby devices.

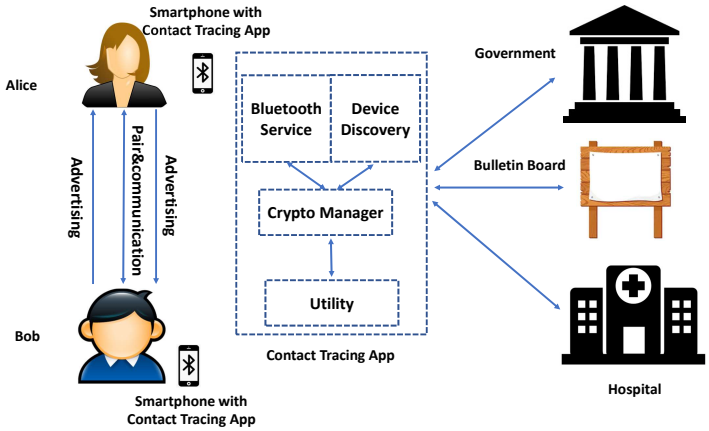

Fig. 3. A diagram of the workflow and architecture of our implementation.

•The Utility component handles regular tasks such as user interface activities, Network IO, to support other components of this app. For example, user may need to download information from Bulletin Board, which is running on the servers maintained by governments or hospitals.

6 Evaluation

We have conducted experiments to carefully evaluate the performance of our contact tracing system. We install the app in a Google Pixel 4 and a Google Pixel 2 smartphones for emulating the interactions of two peoples, and execute the tool for doctor in a PC (Macbook Pro, Core i7, 16GB RAM).

To characterize the latency required by our solution, we measure three phases in the contract tracing app, including meeting, medical treatment and tracing phase. We execute the process including advertising data, receiving and processing packets, for 100 times and compute the latency of each phases. The mean delay and the standard deviation for them are 94ms(49), 4829ms(19), 124ms(4), respectively.

Moreover, we evaluate the time required by the tool for doctor to finish the verification. By running the tool to conduct the verification for 100 times, we observe the mean elapsed time is 515ms and standard deviation is 224.

Note that the mean time for tracing phase includes 1 checking on equation (**??**) plus one verification of a group signature. In the reality, there should be a very large number of checking on equation (**??**) (e.g. 10000) which represents the number of new contacts made by the overall number of new patients. Therefore

we also separately evaluate the time of executing this equation. The mean time is only 72ms. The running of the verification of a group signature should be only a few (e.g. 2 to 3). We can argue that even if there are 100 new patients confirmed each day, and each patient has around 20 to 30 close contact (and overall a few hundreds contacts over the past two weeks), the running time for the tracing phase is still acceptable, and can be completed within a few hours in this extreme case. We have also addressed a practical consideration for this case in the next section.

7 Practical Consideration

In practice, there are some other practical issues behind privacy-preserving contact tracing.

7.1 Cluster Formation

Cluster formation is an important and essential part of analyzing how a disease is spread in the community. The Government can response promptly once a cluster is identified. In Singapore, clusters in foreign workers' dormitories were found and the dormitories were quarantined. In Hong Kong, a cluster was formed in the same residential building and it help scientists to identify that COVID-19 spread through leaked toilet pipes.

In our privacy-preserving contact tracing app, the meeting location for all users are hidden from the Government and the Medical Doctor. We propose that cluster formation should be done after the contact tracing phase. Suppose that Alice is infected and she has contact with Bob and John. If John is not infected, the meeting location of Alice and John should be kept private. If Bob is infected, then the Government can perform normal cluster formation between Alice and Bob (e.g., to see if they work in the same company or live in the same building). The Government can obtain such information from the infected patients.

7.2 Privacy Leakage related to Doctor

In our proposal, we also consider the privacy leakage with respect to the Medical Doctor. For example, if the doctor is specialized in pediatrics, then it is likely that the patient is a child. Therefore, we use the anonymity property of group signature to ensure that such information is not leaked through the doctor's signature on the bulletin board.

On the other hand, it is also possible for the Government to host a COVID-19 information website with access control policy to replace the bulletin board. Only authorized doctors can post in this website and the identity of the doctor is hidden. Then we can replace the group signature and the bulletin board with a standard signature (from the doctor) and the COVID-19 information website. All users should trust the validity of the information posted in this website.

7.3 Improving System Performance

We can see that the number of computation needed in the tracing phase is directly proportional to the number of newly confirmed patients each day. In other to improve the system performance in a country where there may have more than 10000 new confirmed cases each day, we suggest that our protocol can be parametered to the city or state level: we just need to adjust the group signature so that the whole city forms a group (instead of a hospital) and users from the city only needs to check those entries signed by the doctors in the city. In this case, the number of daily new patients can be greatly reduced. It is a reasonable assumption for countries with locked down policy also have banned interstate travelling or even intercity travelling.

8 Conclusion

In this paper, we have proposed a privacy-preserving COVID-19 contact tracing app. It provides privacy-preserving contact tracing capability, in a way that the Government does not know the location and identification of a confirmed patient's close contacts, while these people can be informed of being close contacted with a patient. We use zero-knowledge proving techniques to achieve this nice feature. The cryptographic security has been proven to ensure the privacy is well-preserved and false positive will not be happened. We have also implemented our proposed protocol in Android smartphone. The evaluation result shows that it is practical to be use in even a reasonable large scale.

References

1. T. Altuwaiyan, M. Hadian, and X. Liang. Epic: Efficient privacy-preserving contact tracing for infection detection. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, 2018.
2. Apple Inc and Google Inc. Contact tracing bluetooth specification v1.1. `https://www.blog.google/documents/58/Contact_Tracing_-_Bluetooth_Specification_v1.1_RYGZbKW.pdf`, 2020. Last Accessed 30 April 2020.
3. Apple Inc and Google Inc. Contact tracing cryptography specification. `https://www.blog.google/documents/56/Contact_Tracing_-_Cryptography_Specification.pdf`, 2020. Last Accessed 30 April 2020.
4. Australian Government Department of Health. COVIDSafe app. `https://www.health.gov.au/resources/apps-and-tools/covidsafe-app`, 2020. Last Accessed 1 May 2020.
5. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.
6. D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.

7. S. Brack, L. Reichert, and B. Scheuermann. Decentralized contact tracing using a dht and blind signatures. Cryptology ePrint Archive, Report 2020/398, 2020. https://eprint.iacr.org/2020/398.
8. R. Canetti, A. Trachtenberg, and M. Varia. Anonymous collocation discovery: Harnessing privacy to tame the coronavirus, 2020.
9. J. Chan, D. Foster, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, S. Singanamalla, J. Sunshine, and S. Tessaro. Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing, 2020.
10. C. Gentry. Practical identity-based encryption without random oracles. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, 2006.
11. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
12. Y. Gvili. Security analysis of the covid-19 contact tracing specifications by apple inc and google inc. Cryptology ePrint Archive, Report 2020/428, 2020. https://eprint.iacr.org/2020/428.
13. Israel Government Health Ministry. HaMagen. https://govextra.gov.il/ministry-of-health/hamagen-app/download-en/, 2020. Last Accessed 1 May 2020.
14. M. Li, S. Yu, N. Cao, and W. Lou. Privacy-preserving distributed profile matching in proximity-based mobile social networks. *IEEE Transactions on Wireless Communications*, 12(5):2024–2033, 2013.
15. X. Li and T. Jung. Search me if you can: Privacy-preserving location query service. In *IEEE INFOCOM 2013*, pages 2760–2768, 2013.
16. D. Liao, H. Li, G. Sun, and V. Anand. Protecting user trajectory in location-based services. In *IEEE GLOBECOM 2015*, pages 1–6. IEEE, 2015.
17. Z. Liu, D. Luo, J. Li, X. Chen, and C. Jia. N-mobishare: new privacy-preserving location-sharing system for mobile online social networks. *Int. J. Comput. Math.*, 93(2):384–400, 2016.
18. A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE S&P 2009*, pages 173–187. IEEE Computer Society, 2009.
19. K. Pietrzak. Delayed authentication: Preventing replay and relay attacks in private contact tracing. Cryptology ePrint Archive, Report 2020/418, 2020. https://eprint.iacr.org/2020/418.
20. A. Pingley, N. Zhang, X. Fu, H. Choi, S. Subramaniam, and W. Zhao. Protection of query privacy for continuous location based services. In *INFOCOM 2011*, pages 1710–1718. IEEE, 2011.
21. L. Reichert, S. Brack, and B. Scheuermann. Privacy-preserving contact tracing of covid-19 patients. Cryptology ePrint Archive, Report 2020/375, 2020. https://eprint.iacr.org/2020/375.
22. R. Rivest, J. Callas, R. Canetti, K. Esvelt, D. K. Gillmor, Y. T. Kalai, A. Lysyanskaya, A. Norige, R. Raskar, A. Shamir, E. Shen, I. Soibelman, M. Specter, V. Teague, A. Trachtenberg, M. Varia, M. Viera, D. Weitzner, J. Wilkinson, and M. Zissman. The pact protocol specification. https://pact.mit.edu/wp-content/uploads/2020/04/The-PACT-protocol-specification-ver-0.1.pdf, 2020.
23. R. Schlegel, C. Chow, Q. Huang, and D. S. Wong. Privacy-preserving location sharing services for social networks. *IEEE Trans. Serv. Comput.*, 10(5):811–825, 2017.
24. Q. Tang. Privacy-preserving contact tracing: current solutions and open questions. Cryptology ePrint Archive, Report 2020/426, 2020. https://eprint.iacr.org/2020/426.

25. The Conversation. Coronavirus: South Korea's success in controlling disease is due to its acceptance of surveillance. https://theconversation.com/coronavirus-south-koreas-success-in-controlling-disease-is-due-to-its-acceptance-of-surveillance- 2020. Last Accessed 1 May 2020.
26. The Government of the Hong Kong Special Administrative Region. "StayHomeSafe" Mobile App User Guide. https://www.coronavirus.gov.hk/eng/stay-home-safe.html, 2020. Last Accessed 1 May 2020.
27. C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathe, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, L. Barman, S. Chatel, K. Paterson, S. Capkun, D. Basin, J. Beutel, D. Jackson, B. Preneel, N. Smart, D. Singelee, A. Abidin, S. Gurses, M. Veale, C. Cremers, R. Binns, C. Cattuto, G. Persiano, D. Fiore, M. Barbosa, and D. Boneh. Decentralized privacy-preserving proximity tracing. https://github.com/DP-3T/documents/blob/master/DP3T20%White%20Paper.pdf, 2020. Last Accessed 30 April 2020.
28. S. Vaudenay. Analysis of dp3t. Cryptology ePrint Archive, Report 2020/399, 2020. https://eprint.iacr.org/2020/399.
29. W. Wei, F. Xu, and Q. Li. Mobishare: Flexible privacy-preserving location sharing in mobile online social networks. In A. G. Greenberg and K. Sohraby, editors, *IEEE INFOCOM 2012*, pages 2616–2620. IEEE, 2012.
30. R. Zhang, J. Zhang, Y. Zhang, J. Sun, and G. Yan. Privacy-preserving profile matching for proximity-based mobile social networking. *IEEE Journal on Selected Areas in Communications*, 31(9):656–668, 2013.