GeauxTrace: A Scalable Privacy-Protecting Contact Tracing App Design Using Blockchain

Tao Lu
Electrical and Computer Engineering
Louisiana State University
Baton Rouge, USA
tlu4@lsu.edu

Fang Qi
Dept. of Computer Science
Tulane University
New Orleans, USA
fqi2@tulane.edu

John Ner
Electrical and Computer Engineering
Louisiana State University
Baton Rouge, USA
jner1@lsu.edu

Tianqing Feng
Electrical and Computer Engineering
Louisiana State University
Baton Rouge, USA
tfeng3@lsu.edu

Brian Cunningham
Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana and Champaign, USA
bcunning@illinois.edu

Lu Peng
Dept. of Computer Science
Tulane University
New Orleans, USA
lpeng3@tulane.edu

***Abstract*—Contact tracing is the approach to identifying physical contact between human beings using a variety of data such as personal details and locations to discover the potential infection of diseases. Since the outbreak of the COVID-19 pandemic, contact tracing has been used extensively to quarantine the people at risk to stop the spread. Moreover, the data collected during contact tracing are typical spatiotemporal data, which can be used to study the disease and discover the spread pattern. However, both traditional labor-intensive and modern digital-based approaches have limitations in terms of cost and privacy concerns. In this paper, we proposed GeauxTrace, a Blockchain-based privacy-protecting contact tracing platform, which separates private data from proof of contact. Sensitive data collected by the front-end app via Bluetooth-based methods are stored locally, and only the proofs of contacts are uploaded onto the immutable private blockchain, which forms a global contact graph at the backend. Our approach not only enables multi-hop risky users to be notified but also reveals the infection patterns via the global graph, which could help study diseases and assist the policymaker. Our implementation shows the feasibility of the proposed platform in real-world scenarios and achieves the performance of 20-30 user requests per second.**

***Index Terms*—Contact tracing, Big data infrastructure, Blockchain application, Privacy protection**

I. Introduction

Contact tracing is the process of identifying, assessing, and notifying people who have been exposed to a disease to prevent onward transmission [8]. Historically, contact tracing has been widely used to investigate and prevent infectious diseases and has been more important to the recent COVID-19 pandemic due to the lack of vaccines and medicines. Moreover, the data collected are typical spatiotemporal data [7], which can be used to study the disease and discover the spread pattern using big data mining, and the industry has been long for a flexible infrastructure to collect and manage these data.

Traditionally, contact tracing is labor-intensive, which relies on people to recall their contact in the past days or even weeks. The accuracy and completeness have been the bottleneck and have harmed the effectiveness. Nowadays, technologies have been used to resolve the limitations and process the data in digital methods. The data can be collected via a variety of methods such as Bluetooth, GPS, CCTV, financial transactions, etc., and then processed at the backend. These processing approaches can be roughly divided into two categories, namely, centralized and decentralized. Centralized approaches rely on a trustworthy entity (usually the government or authorities) to collect and analyze data on a centralized server. These approaches maximized the accuracy and completeness but brought more concerns about security and privacy. On the other hand, decentralized approaches rely on personal devices to detect and trace individually. A good example is Google and Apple's exposure project [1], which relies on the smartphone's Bluetooth module to discover the contact and compare it against a downloaded anonymous list of infected people.

According to our observation, both directions have their weak points. The centralized method arises concerns about privacy issues that sensitive information could be abused or leaked. Further, even though the operator could be under tight regulation, there is still the risk of a single point of failure. On the other hand, decentralized methods are privacy preserved but they also hide important information that could be useful. For example, there is no way to have an overall picture to study the spread of diseases. Furthermore, the secondary contact, who is in contact with the primary contact of a positive patient, cannot be notified because of lacking multi-layer tracing.

In this paper, we propose GeauxTrace ("Go-trace"), a blockchain-based privacy-protecting platform to overcome these challenges. We observed that personal information, such as identity, location, travel history, etc., is not relevant in contact tracing; what matters is the de facto physical contact that happened between different people, which reveals the infection mechanism. Our approach uses a Bluetooth-based detection mechanism at the front end and hides the identity using a random ID. Then the client submits a proof of contact, which is a transaction signed by two sides of the physical contact, to our server where the transaction is validated. If the proof of contact is valid, it guarantees that a real contact happened, and such information is recorded by the blockchain.

At the backend, valid contact data are traced and used to generate a global contact graph. Once a user reports a positive diagnostic result, our server can trace the graph back up to 14 days and inform the users at risk to take action. More importantly, since we have a global view, users multi-hops away from the patient can also be discovered and notified by tracing the graph. In this way, we keep the private information from leaving personal devices but generate the overview contact history anonymously and thus provide more useful information than the above works without sacrificing privacy. In addition, our platform is open to third-party applications and provides connectivity to support many other applications. We have application programming interfaces (APIs) for other front-end detection and self-test tools to report contacts and we offer access with permission to our private blockchain and servers for analytic usage. Our contribution can be summarized as the following.

- Implemented a Bluetooth-based contact detection application (mobile App), which can hide private information but prove a physical contact happened using cryptographic signatures.
- Proposed a blockchain-based privacy-protecting data storage platform, which can gather anonymous contacts and generate a global contact graph.
- Offered scalable and flexible connectivity to other upstream detection applications via APIs and downstream analysis tools through access with permission.
- Implemented the whole system and evaluated the performance in real-life scenarios.

II. Background and Motivation

In this section, we introduced the background of contact tracing, blockchain technology, related works, and motivation.

A. Contact tracing and related works

Contact tracing is the method that monitors the data such as personal details and locations to discover the potential infection between human beings. Historically, contact tracing has been widely used to investigate and prevent infectious diseases and has been more important to this COVID pandemic due to the lack of vaccines and medicines. Traditionally, contact tracing is labor intensive, which relies on people to recall their contact in the past days or even weeks. The accuracy and completeness have been the bottleneck and harmed the effectiveness. Nowadays, modern technologies have been used to resolve the limitations.

Contact tracing has been proven to be effective in studying and controlling diseases [6]. Especially in the combat against the recent COVID-19, a lot of efforts have been put into digital contact tracing to slow down the infection. Digital contact tracing methods are collecting data via a variety of methods such as GPS, CCTV, financial transactions, etc. Among them, the most used approaches are smartphone-based ones using Bluetooth [5]. In this paper, we choose this well-accepted approach because that offers the best availability to common people with minimal additional cost. Since almost all smartphones today support the Bluetooth module that can be used as a detection method, all we need from a normal user is a piece of app.

In terms of how the data are collected and processed, these approaches can be roughly divided into two categories, namely, centralized and decentralized ones. Centralized approaches rely on a trustworthy entity (usually the government or authorities) to collect and analyze data on a centralized server. These approaches maximized the accuracy and completeness but bring more concerns due to the following reasons. First, centralized approaches have a huge social cost and thus only work well when the number of cases is low. Second, other government-based approaches bring concerns about security and privacy. For instance, the Israeli government tracks the mobile phone data of suspected COVID-19-infected persons, the South Korean government stores personal data of known patients in a public database, Taiwan's medical agency keeps track of patient's travel history, and so on [4]. These limitations harm the effectiveness and the citizen's willingness to cooperate.

On the other hand, decentralized approaches rely on individuals to detect and trace on their own devices, in order to address privacy issues. A good example is Google and Apple's exposure project [1]. This project installs a piece of software on users' smartphones, which uses the Bluetooth function to discover the contact and save it locally. Then every day, an official authority will broadcast a list of IDs of patients who have been diagnosed positive. After downloading the list, the software compares it against the local physical contact records and checks if there is an overlap. If matching is spotted, the user is considered exposed and got notifications from the software. This method and many other similar ones try to protect privacy by storing the record locally and not sharing the local contact history with any external sources. The drawback is that it does not gather the information together so it can only track one hop contact. In another word, it has no way of having an overview of the infection pattern nor being able to notify any user-in-risk two hops away (secondary contacts).

There are also blockchain-based solutions addressing how to safely store private data on the blockchain. For example, [3] uses Self-Sovereign Identity (SSI) proofs and [11] uses DAG-based duo-chain architecture respectively. These approaches utilize complex cryptographic methods to encrypt personal data and tried to store them more safely. However, our approach solves privacy issues in another way. Instead of saving all the information, we separate the contact information from the personal data and only the proof of the contact using signature is stored on the blockchain, which further improves privacy. In addition, we maintain a graph structure that can be easily traced and compatible with authorized connections to third-party applications. We observed that in the contact tracing study, the identity is not actually relevant. What matters is the physical contact relation between patients which reveals the infection mechanism. Our blockchain-based approach hides the individual identity but preserves the contact history and thus provides more useful information than the above works.

B. Blockchain

Blockchain is a trustless distributed system that allows data to be stored in an anonymous and immutable way. From a historical point of view, blockchain is the technology that implements Bitcoin, which is the first peer-to-peer anonymous payment system (cryptocurrency). Since the invention of Bitcoin, the usage and the meaning of blockchain technology has been extended tremendously. For example, the Ethereum project introduced the smart contract, which allows developers to design complicated functionality over the security fundamental of blockchain. Meanwhile, several projects are trying to implement a variety of consensus mechanisms to improve efficiency. Some projects are exploring new architecture such as Sharding to resolve the performance bottleneck. Some projects are working on the second layer to offload the burden of the blockchain without losing security. Nowadays, blockchain has been a disruptive technology that has several variants and extensively serves many industries.

C. Privacy

The uniqueness of blockchain is that it can protect privacy without sacrificing security. Compared to traditional data storage, blockchain relies on cryptographic methods to achieve anonymity and security. To interact with the blockchain, users need to create accounts which are the basic entity on the blockchain. When an account is generated, the user is given a pair of keys, which are called a private key and a public key. The private key is the most confidential and should not be exposed to anyone other than the owner at any time. The public key, however, can be safely shared with other blockchain users and is also used to derive the account identifier address. The account address, which looks like a string of random characters, is the only identifier on the blockchain. In this way, the information on the blockchain is purely anonymous and the user does need to reveal any identity in order to interact with the blockchain. Moreover, any user can easily generate as many as accounts they want to protect themselves from being traced.

D. Security

The data stored on the blockchain is also secured from tampering and mutation. Due to the nature of asymmetric cryptography, the private and public key pair offers the essential feature of the blockchain, verifiability. When users want to announce something and want to convince others that it is indeed their intention, they can use their own private key to generate a digital signature. Then validators on the blockchain can easily verify that the information is indeed from the sender using his public key. In this way, any imposture and counterfeit can be easily detected and rejected by the blockchain, which ensures all the transactions are valid. Meanwhile, blockchain technology leverages the hash function to ensure the integrity and correctness of the historical record. The hash functions are a collection of functions that can intake any length of input and generate a fixed length of digested information. Essentially, the hash function provides blockchain an easy way to check if the data has been tampered with. On the blockchain, each block is hashed and the hash value is used to link the next block, which results in a chain of traceable hash values, often in a tree structure. By checking the hash values, any participating node can spot the tampered block and reject it, which in turn guarantees the correctness of all the history. Toidentifierh the validity of the new block, we get a verified result in a trustless world.

III. Design and Implementation

In this section, we present the design details of our system. We start with the overall design and then reveal the implementation and rationale of subsystems, namely the private blockchain, the client, and the server. More importantly, we discuss how the subsystems work as a whole using an example and the extensions of the platform.

A. Overview

From a top-level perspective, our system can be divided into three subsystems: namely, the Private Blockchain and validators, the client app, the Geaux (sounds ”Go”) Server and the notification service. Figure 1 shows an overlook of the whole system and the data flow. In general, the private blockchain is where data are stored, and it serves as a secure and privacy-protecting database. Validators are working together with the private blockchain to enhance the security. The client app is where Bluetooth-based contact detection is performed and where users get notifications. The Geaux Server sits in the middle and is in charge of gathering the data to generate the global graph and handle notifications and respond to queries. In the following sections, we introduce the details of these subsystems and how they interact with each other.

B. Private Blockchain

Our Private Blockchain subsystem is the backbone of the whole platform, which serves as the data storage. In general, a blockchain system is composed of several nodes connecting with each other and forming a distributed network. These nodes oversee processing the transactions independently and maintaining consistent data using a consensus algorithm. There are many options to build a blockchain network. For example, we can choose the data structure, the degree of permission, and the consensus mechanism. In the following sections, we discuss our design rationale and implementation details.

1) Choosing the model: First, we need to choose the blockchain model. There are several options to build a blockchain network. For example, Bitcoin uses an unspent transaction output (UTXO) model, which can easily track the source of each transaction but it fits more appropriately for the

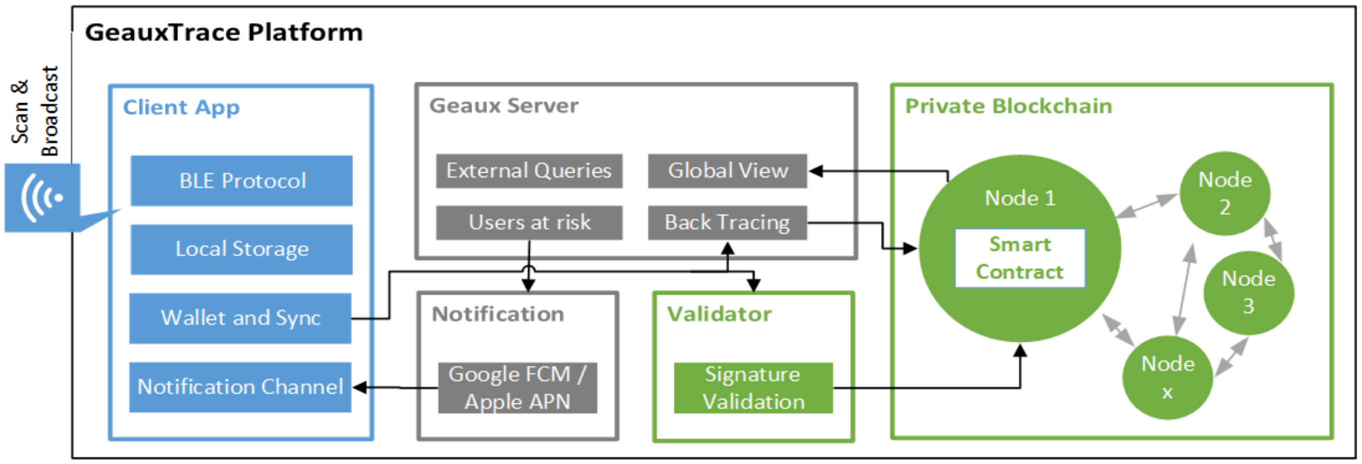

Fig. 1. Overall design of GeauxTrace platform

use of cryptocurrencies. On the other hand, Ethereum is using another model that saves the state of each account and keeps updating it by processing the transactions. The big advantage of having a state is that it supports the smart contract that we can program the logic according to the state change.

We decide our platform must be compatible with smart contracts because we are aiming at flexible functionality. Using smart contracts enabled us to deploy automated, programs and store the contacts in the smart contracts' storage. Given all the available smart contract compatible options such as Ethereum, Hyperledger Fabric [2], EOS [10], etc. We decided to build it based on Ethereum. It is the most well-supported ecosystem because of its largest developer community and the most supported tools. Choosing Ethereum retains the potential for connecting with other tools in their most popular ecosystem.

2) Permissoned or permissionless: Our second decision is to choose between the permissioned and permissionless models. The permissioned (private) blockchain, means we can control who can join the system. The permissionless (public) blockchain, means all the blockchain history, though encrypted and anonymous, is open to the public. We chose our design as a permissioned model due to the following reasons. First, there is no need to expose health information to all non-related individuals even though they are anonymous. Our object is to provide a global view of contact tracing for research purposes and only notify the related people at risk. Second, running the permissionless model requires much more resources on the consensus mechanism, which in turn limits the performance of the system. Our choice is a balance between performance and openness without losing security and compromising privacy.

3) Choosing the consensus: The last key point to determine is choosing the consensus algorithm. Such algorithms determine how the nodes in the blockchain network agree with each other. There are three dominating and time-tested algorithms, which are Proof-of-work (PoW), Proof-of-Stack (PoS), and Proof-of-Authority (PoA). PoW is known as the mining algorithm of Bitcoin and is also criticized as a tremendous waste of power. PoS is more environment-friendly than PoW but also brings vulnerability. These two consensus algorithms are a better fit for the permissionless setup and are barely beneficial in our private blockchain. Therefore, we pick one of the PoA algorithm Cliques as our consensus algorithm, which is already proven to be effective and efficient in Rinkeby and Goerli systems.

4) System Setup: The blockchain system is normally composed of a series of nodes, which are in charge of processing the transactions and appending them to the local history blockchain. As a prototype, we set up our blockchain nodes using three virtual servers on a cloud. In the production environment, the number of nodes can be increased without sacrificing performance. Actually, the more nodes joined the system, the more secure the system as a whole because more faulty nodes would be tolerated. To configure the blockchain as a private one, a secret network id is chosen and shared only among participating nodes, so that other people cannot connect to our system without knowing it. In addition, we enhanced the security by using the IP filter to block any unauthorized access. Our nodes are assigned a static and known IP address that is registered on the whitelist, which allows these nodes to discover and connect with each other. These are done by configuring the genesis file and the node software before establishing the system.

One thing worth noting is the decentralized nature of our platform. As the only entity testing this system, our team controlled all three nodes, which makes the platform looks centralized. However, in real-world scenarios, each participating entity, such as medical authorities, government, or research institutes, should set up their own nodes and never expose their private keys. In this way, they are governing and protecting the system collaboratively, which makes the platform decentralized. They can vote to add or remove signers democratically and cooperate to protect the system. Any attack attempting to alter the blockchain record requires control of more than half of the total nodes. Therefore, it is also encouraged for each participating entity to set up multiple nodes to further strengthen decentralization and security.

Blockchain nodes are working independently to compose a decentralized network. To guarantee data consistency between the nodes, the consensus mechanism must be deployed. Before the whole system starts to run, each node needs to create an account in order to participate in the data exchange and validation. During the account creation, a private key will be issued and should always be kept secret by the owner. To configure the blockchain as private, separate accounts are created on the nodes individually and they will serve as the validators for the PoA consensus. Later, these validation nodes are collecting transactions sent by the clients and package them into blocks with valid signatures. Only the block validated by the majority of authorized validators is considered valid and can be appended to the blockchain history. Meanwhile, we deployed an auto-pause script that is monitoring the transaction pool constantly. When the pool is empty, the production of blocks is paused, so that we can avoid generating unnecessary blocks. After all these steitse configured properly, the network is ready to process any transactions.

5) Smart contract: Smart contracts, in general, are programs running on the blockchain, which handle the state of all the accounts and define the functions to execute when called. Our smart contracts are first coded using a specific high-level language called Solidity and then compiled into Ethereum-compatible bytecodes. Finally, these codes are deployed onto our private blockchain during the system setup. Once successfully deployed, it can be called by sending proper transitsns from the client or the server. For example, the client can send proof of contact between two users to update their storage. Also, the server can query the smart contract to get the stored contacts of a user. By checking these results, the server can do additional jobs accordingly, which we will introduce later in the server section.

Our smart contract uses hash tables as storage where the contacts of each user are stored. The biggest advantage of using a hash table is the speed because the lookup time is O(1) in complexity. It also defines all the functions such as how to validate the transaction and update the storage, and what data should be returned to the server.

6) Faucet: To interact with the blockchain, any account needs some initial funds to be able to send transactions. This is because the original Ethereum requests some tokens to be used as a fee, which is also known as the gas fee. The existence of a gas fee requires the user to pay some token when transactions are sent. So, we set up a faucet service to distribute these tokens to make sure the user has enough balance. However, due to security reasons, we limit the total amount of tokens a specific account can request in 24 hours to minimize the abuse.

7) Validation: The contact log, originally generated by the smartphone client, is validated before they are able to be recorded on the blockchain. After receiving the message, the server will validate the signatures of the contact log and send it to the smart contract. The purpose is to guarantee two aspects: 1) All the contacts are real, which means signatures from both counterparts are valid. 2) A user is only updating his own account, which protects against the attacks. Once the blockchain got the transaction, the function defined in the smart contract will be triggered and the log will be stored on the blockchain as a historical record. If it is the first time that a user is added to the blockchain, specific storage space will be created for this user. If this is the follow-up update, then the contact log will be appended to the user's existing storage space. According to the suggestions from CDC, we are only tracing back up to 14 days, therefore any data older than that can be removed to save storage space

C. Client

1) Overview: The client is the smartphone application where contacts are detected, and data are collected. Our approach relies on the built-in Bluetooth Low Energy (BLE) module of smartphones, and we designed GeauxTrace (pronounced "Go-trace") app for both the Android OS and iOS systems. As shown in Figure 1, the GeauxTrace app is composed of four major parts: the BLE protocol used in contact detection, the Local Storage module, the wallet and synchronization module which handles the interaction with blockchain, and the notification channel to receive notifications.

From an overview, once a new user registered on our platform, the app will generate a unique UserID randomly, which is the only identifier used in our platform. When users start the app, the Bluetooth protocol starts scanning and broadcasting to discover the other users nearby. Our data in the air are encrypted and we set an app ID to guarantee that only our app can decode it. Once another app is detected, they exchange the UserID and mark the timestamp. The scanned results are temporarily saved in local storage. We checked the distance by measuring the RSSI of Bluetooth and monitored the contact duration.

If a physical contact meets the criteria (15 minutes and 6 feet) suggested by CDC [9], we consider it valid physical contact. Our app will periodically update the new contacts to the backend blockchain by submitting proof of contact happened. And if risky contact is detected, the user will get notified through the app. GeauxTrace is decentralized, with user privacy as the top priority. No personal data is required to start using the app, nor does the app record any personal data when using it, such as sensitive GPS locations. In the following sections, we introduce the key modules, design details, and how the app works in detail.

2) Contact log data structure: The contact log is the information that records the fact of physical contact. It includes the counterpart's identifiers, the timestamp when it happens, the time duration, the estimated distance, and the proof of contact. Such data are generated by the Bluetooth protocol and then saved locally in the app until they are synchronized with the server in order to be processed on the blockchain. To make our design more secure, the data is encrypted and saved in our app's private storage, which prevents other malicious apps from accessing our data. We introduce these concepts below in detail.

Identifier: The identifier (ID) in our protocol is called userID, which is a 128-bit long integer. In order to conceal the user's identity, we did not use any sensitive information that could potentially harm the user's privacy, such as Bluetooth MAC address, GPS locations, Apple ID, or Android identifier about the device. Instead, we generate a random userID for each user individually, when they register. Such ID is long enough so that even though it was generated randomly, there is little chance that two users are going to share the same id, causing a collision.

Proof of contact: The proof of contact is the pair of digital signatures from both ends of a contract, which is used to protect any faking. When the user installed our app, they will be granted a pair of public and private keys. Then the user will generate a signature of their ID to prove the ownership of his ID and include that in the Bluetooth broadcasting payload. When a contact happens, the receiver will receive the message and include the signature in their contact log together with another signature signed using his own key. Then both signatures are validated by the blockchain later to prove the ownership of both IDs. Thus, we can guarantee a contact did happen between two real IDs and was detected by the device.

Timestamp and Duration: The timestamp marks the time when the contact happens, and the duration indicates how long it goes. The timestamp relies on the smartphone to provide the time from OS. The duration is checked by checking the signals periodically. We perform the scanning with a fixed time interval. So, if two consecutive scans detect the same target, we consider the contact still in process. To determine how long a contact should be considered risky, we followed the 15 minutes rule suggested by CDC, but this is subject to change.

Estimated Distance: The contact distance is estimated by measuring the Bluetooth signal strength, aka Received signal strength indication (RSSI). Ideally, the signal strength is decreasing as the distance increases. Therefore, we can estimate the distance by checking the signal at the receiver side. It is worth noting that we do not need precise distance because 1) There is no way to measure accurate distance only using the Bluetooth-based method. 2) An estimated distance satisfies our need. The first point is due to the fact that Bluetooth can be easily interfered with by walls, doors, and other obstacles. And the latter is because we only need to determine if a contact has a risk of disease spreading. Therefore, we set up an RSSI threshold to estimate the distance of 6 feet as suggested by the CDC.

3) Bluetooth protocol: The Bluetooth protocol plays a key role in contact tracing. It defines how to detect contacts, and generate and exchange contact logs. Our Bluetooth protocol defines the data structure of the Bluetooth payload and handles both the signal sending and receiving. Once the app is running, it broadcasts and listens simultaneously via the smartphone's built-in Bluetooth module, and an App ID is used to distinguish our detection from other Bluetooth services. When a device discovers a potential contact, it will set up a private channel and exchange the information to generate proof of contact. Later, this information is gathered in the daily contact logs and uploaded to the blockchain. Figure /refbluetoothProtocol shows the procedure of the Bluetooth protocol and the data structure of the payload. We explain the details below.

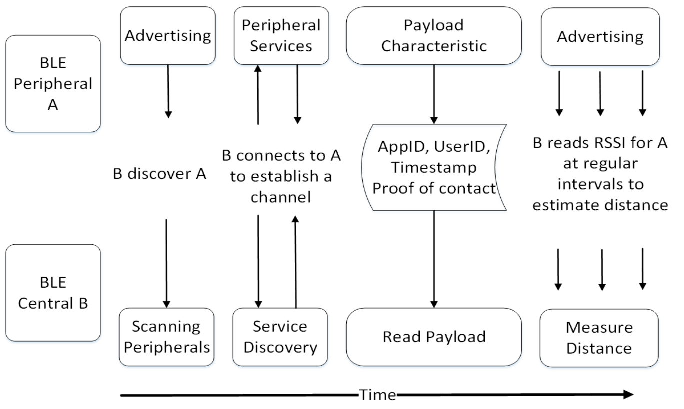

Fig. 2. Bluetooth protocol and lifecycle

Payload data structure: The information in the air, aka the Bluetooth payload, is composed of three parts. The first eight bytes are the Application ID (AppID), which is used to distinguish our app from the others so that only our app can decode the payload correctly. Meanwhile, the AppID can also be used to connect our app with third-party apps to provide more features. For example, we successfully connect our app with other smartphone-based self-testing applications to allow users to test and report the diagnostic results. The next section in the payload is the UserID, which is the anonymous identifier to locate the user in our system. Next, follows the timestamp which indicates the current time. The last section is the proof of contact which contains the signature of the sender and his public key, which can be used to validate the ownership of his UserID.

Scanning and Broadcasting: When the app is started, it broadcasts and scans simultaneously and periodically. Broadcasting is sending out the information so that it can be decoded by another smartphone running our app, which is also known as the peripheral mode. Scanning is actively checking the Bluetooth signal in the environment and filtering out the ones from our app, which is also called central mode. Our scanning and broadcasting are not continuous. Instead, they are performed periodically with a fixed scan interval and broadcast interval. This is because we can tolerate a small-time delay when contact is happening, and more importantly, we can reduce power consumption. Obviously, the more frequent Bluetooth scanning is performed, the more accurate the contact could be detected. So, the user can trade-off between accuracy and power consumption by changing the configuration in the App.

4) Detection Procedure: Figure 2 describes a full cycle of BLE detection. GeauxTrace will automatically advertise and scan according to the state of login and Bluetooth status of the device. A device scan is initiated up to eight times per minute to discover other devices running the same protocol. Device A running the BLE in peripheral mode means it was broadcasting its signal and ready to be discovered. Device B in central mode is actively searching for neighbors and trying to set up the channel. When A and B are in the range of each other, B is able to discover A, and a channel is set up through the peripheral and discovery service.

When a device is discovered, the app will check if the target device offers the same service protocol and the OS of the device. If both are correct, a connection between the two devices will be established to exchange payload data. Payload data is obtained from the target device once the payload characteristic has been discovered and received. Meanwhile, the RSSI is read periodically when the connection is set up, which is then compared to the threshold value to verify if the data should be discarded or stored in the contact list. If criteria are met, we check the target UserID and timestamp to determine the contact duration. if two consecutive scans detect the same UserIDs, we consider the contact still in process. Until the contact discontinues, the duration is calculated based on the recorded timestamp, and the contact log is generated and stored in the persistent memory of the devices.

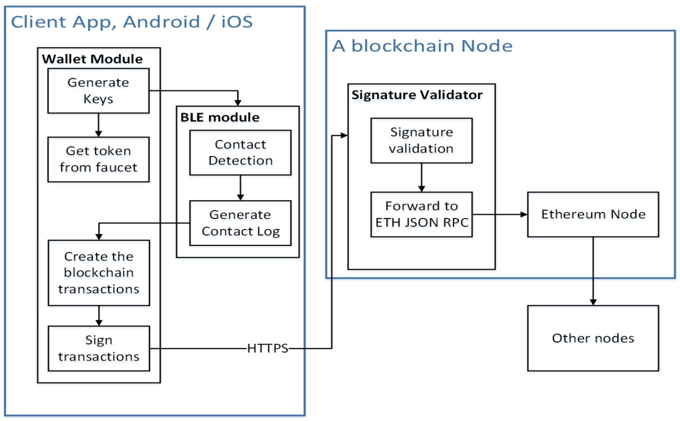

Fig. 3. Client interact with Blockchain

5) Wallet and Blockchain interface: To send transactions to the blockchain, a piece of software is needed to send the transactions and control the account, which is called a wallet in the blockchain context. Even though our project does not need any cryptocurrencies, we use the same term for the submodule that composes the blockchain transaction and generates the signature. Meanwhile, the wallet will hold the tokens requested from the faucet and pay the gas fee as we mentioned above.

Figure 3 shows the interaction between the mobile app and the blockchain. When a user installs our app, the built-in wallet module will automatically generate a pair of public and private keys. The private key is saved locally and will never be exposed to the external and the public key is the blockchain account address. When it is time to synchronize with the blockchain, the wallet will compose the transaction with the recorded contact log and sign it using its private key. The signature is sent together with the transaction to guarantee the ownership of the account. By validating such signatures, we guarantee that a user can only update their own account. Whenever the user's balance runs low, the wallet will also send a request to the faucet server. As long as the user did not request an excessive amount, they will be issued tokens automatically.

In terms of the interaction with blockchain, it has been well-known that using a wallet is complicated and cumbersome. Our design keeps it to the minimum that a normal user does not need to have any knowledge of the blockchain and cryptocurrencies. We hide the wallet under the normal App appearance so that a normal user will not even realize the interaction with the blockchain. The generation of keys, the signature, and the requests for tokens are all automatically done in the background to reduce the user's hurdle.

6) User interface: GeauxTrace is designed to be easy to use and requires little interaction. Users need to sign up for an account using an email address, which is only used for password recovery. After login, users can keep the app in the background, and the app will work automatically without any further interference. It will detect the contacts around the user to generate a local contact report and synchronize with the server every several hours (typically 24 hours). If a user is diagnosed as positive, the user would need to self-report by using the report function in the app. Once the server received the report, a back tracing is performed in the contact history to find out users at risk, and a notification is sent anonymously to these users to give them a warning.

We use Firebase by Google to handle account authentications and notification services. All the information that users used in creating the account, such as username(email) and password are hashed, encrypted, and saved to the database on Firebase. Every time a user logs in, the login information is encrypted and compared to the information on the Firebase server, therefore no password is revealed during the network transportation. Meanwhile, Firebase handles the notification service by issuing a unique FCM(Firebase Cloud Messaging) registration token to each user. The FCM token is created and fetched from Firebase servers during the app installation. Our server also uses such tokens to identify the phone(app) that should be anonymously notified when they have been in contact with a positive user. The Firebase Cloud Messaging service (FCMs) and Apple Push Notification service (APNs) handle this process, and the user receiving the message is instantly notified.

During our design, there are several considerations that aim at protecting privacy. For example, it was possible that a user can make up a username instead of using a real email if he gives up the password recovery because we did not rely on any real-world personal information. In addition, a user can only be notified at most once every 24 hours. This limitation not only protects users from flooded notifications but also conceals the social identity of contacts. For example, once a user got a notification, it simply indicates that one of the contacts in the last 14 days is reported as positive. Neither the contact time nor the contact's id will be revealed. It does not necessary to be the contact in the last 24 hours, therefore it is unlikely that the person could be known. The user will always be notified as soon as the first risky contact is reported. But if multiple notifications are triggered, they can also configure how often they want to be notified in their app, to have a better user experience.

7) Storage security: : Since it is expected that devices will move in and out of range, sometimes permanently, the information received during scanning and connection is stored in a temporary cache. The cache is purged and refreshed once the information on the device has been out of range for at least an hour. The data is deleted once the user signed out of their local account on the App.

The data required to create an account and login does not contain personal information. Our app generates UserIDs that are not linked to the user's personal data. Location information is not used in our application. The user identification information and data are safely stored in the phone's persistent data through Apple's Core Data API. A data model is created, and it has minimal data structures for simple read/write. Any uploaded data by our app is hashed and encrypted. The system handling the uploaded data uses decentralized and blockchain techniques to protect it from potential hacks and ensure user privacy.

On the cloud side, contact data has an expiring time of 14 days, according to the suggested threshold by CDC. Expired data are removed from the account by the smart contract without interaction and will not be used in the back tracing algorithm. To meet the privacy policies, a user can also request the deletion of his account from the blockchain. Due to the decentralized nature of blockchain, a such request needs to be approved by the majority of node runners. And since our setup is permissoned blockchain, where we can assume the minimal malicious behavior of authorized anticipates, this is acceptable to our needs.

D. Server

We set up our server to handle the back-tracing and generate a global contact graph. Whenever a patient is reported positive, it can specify the users at risk and generate notification requests to the FCM service and APN service. To reduce the network latency, the server is set up on the same machine with the blockchain node. It is possible that multiple servers can be set up and a network load-balancer should be used to balance the traffic load. And the number of servers should scale up as the number of users increases over time.

1) Back tracing: Figure 5 shows the logic of back tracing and the generation of notifications. When the blockchain receive a contact log, it will first check if the owner has reported as positive. If the account owner is negative, the server will only update the account status by forwarding the transaction to the smart contract. After the smart contract is executed, the account's contacts stored are updated. There is no point in doing back tracing for a negative user automatically, but our server can still do back tracing if necessary, which can be done via API for research purposes.

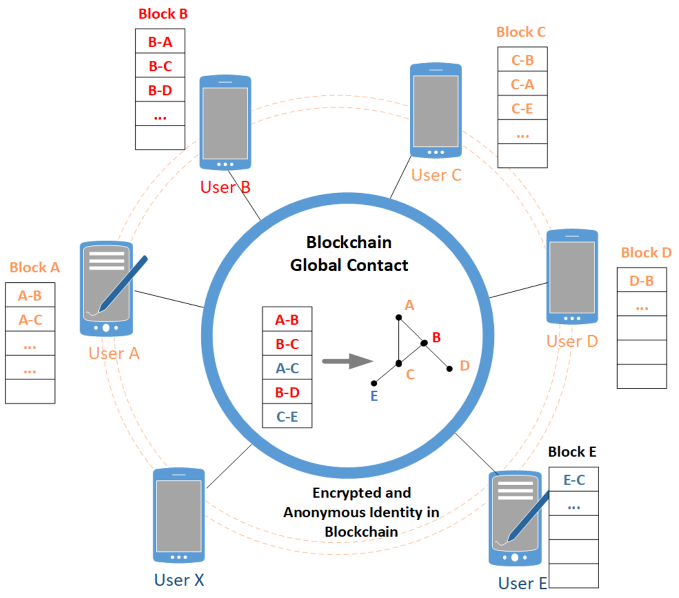

Fig. 4. Interaction Illustration

Once the server receives a positive report, it will perform the back-tracing algorithm to find out who should be informed. The process starts with pulling up the contact history from the blockchain by sending a query to the blockchain node. Since such a query does not modify the storage of the blockchain, aka view only, it was not executed by the EVM. It does not need to be mined nor append to the blockchain history. A view-only function is programmed explicitly in the smart contract so that it can be handled by the blockchain nodes directly. These nodes will query their local blockchain history to pull up the contact log in the last 14 days and return it to the server. This is also the reason our server is set up on the same machine as the blockchain node because it reduces the data transfer time. However, splitting these two roles into separate machines is feasible but adding some network delays as a trade-off.

After receiving the data from the blockchain node, the server will do a depth-first search to gather all the possible contacts' UserID. Once these IDs are collected, it can search for the FCM token bonded with these ids and send notification requests to the Firebase service, which later sends out notifications to the corresponding smartphones. Meanwhile, the traced data are saved in a graph structure in the server node, so that it can be used for data mining. The server can also respond to external queries via APIs so that external user does not need to download the data. In the section below, we explain how this works use through an example.

2) How the system works: Figure 4 shows how the subsystems described above work together. Suppose users A, B and C have a meeting together, their app should be able to detect each other and generate their local records independently. User A's app will record a contact of A-B and A-C; user B's app will record B-A and B-C and so on. Similarly, if there is private contact between B-D and C-E, their client will record

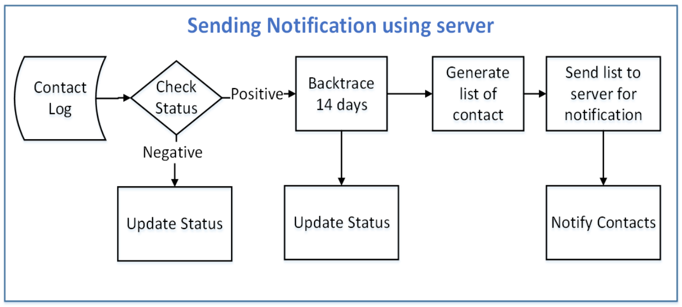

Fig. 5. Back tracing and notification logic

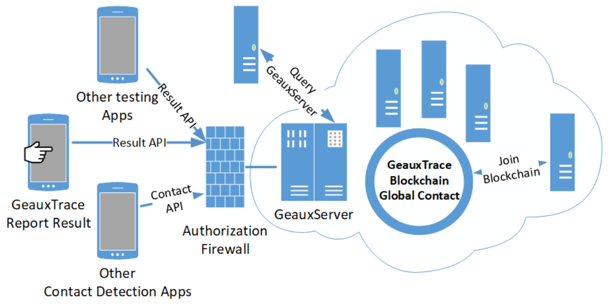

Fig. 6. APIs and Connectivity

such info as well as shown in the Figure. Then these clients will send update messages to the blockchain independently, therefore the contacts will be stored in the different blocks. For example, block A will only store the contact log submitted by user A and so on. After receiving the updates, the server will generate the contact graph like the one in the middle. A through E will be the vertices identified by their UserIDs and the edges indicate the contacts between the users, such as edge AB, BC, etc.

Later if user B is diagnosed as positive, he should submit his result by reporting via his app. Once the server got the positive update, it will pull up the history from the blockchain and get B-A, B-C, and B-D from Block B (marked red) and locate vertex B as risky (red vertex) in the graph. Then we can do the breadth-first search starting from the risky vertex B. If vertex A's contact is not in the server's cache, it queries the blockchain to get A-B and A-C. Similarly, by query contacts of C, we got C-A, C-B, and C-E.

We define the vertices x hop away from the nearest positive vertex as the x-level risky vertices. Therefore, vertices A, C, and D are level-1 risky (in orange), and vertex E (in blue) is level-2 risky. Finally, we can send different notifications to these risky users and give them advice on how to react accordingly. For example, the level-1 risky users, aka primary contacts, need to be tested and quarantined. And level-2 risky users, aka secondary contacts, only need to monitor symptoms and reduce exposures.

Treating the users at different risk levels will allow us to control the scope of notification and give users different suggestions, which truly help them to prepare more appropriately, thus could potentially improve the effectiveness and user experience. Meanwhile, from a global perspective, it gives us a better understanding of how the disease is spread as time goes by and reveals more information compared to related works which merely send out notifications passively.

3) Cybersecurity Considerations: Although we mentioned our authorized nodes can be trusted to some degree, we cannot trust external entities. We implement several techniques to protect our server from attacks. Firstly, we set up a firewall to filter the whitelisted IP addresses. Our servers are assigned static IPs that are registered on a whitelist. Only queries from these permission-ed IPs are accepted by the Blockchain, which protects against unauthorized access. Secondly, all the communication between the servers and clients is encrypted by the HTTPS protocol, which minimizes the possibility of middleman attacks. In addition, any transactions submitted by the client will be accompanied by a signature generated by the wallet on the mobile device. This method guarantees a user can only modify their own record because invalid signatures can be easily detected, and such transactions will be rejected. Finally, we limit the rate of submitting results from a specific client, and the requests will be rejected if exceeding our threshold. This protects us from DoS attacks and any malicious abuse of the client app.

4) Scalable Connectivity: Our platform is designed to be flexible and scalable, which means it can connect to other tools from both upstream and downstream sides. To the upstream, we opened our platform to other self-testing and/or contact tracing mobile apps. We defined our APIs, which specify what data and in what format a contract should be submitted to our blockchain. Additionally, we also reserve optional metadata which can be customized for specific usage. The left part of Figure 6 shows our two APIs, which are used to submit contacts and diagnostic results respectively. Any third-party app can submit its data using either or both APIs, after being issued an authorized token to pass our firewall.

On the downstream side, we open our system in two ways. One offers an API to access the graph information from the GeauxServer. We can authorize a third party to query our server and use it as a normal database to research the pandemic. The other way provides the opportunity to run a node composing the private blockchain. We welcome properly authorized entities to join our system and set up their nodes so that they have full access to the blockchain record. In that way, they are free to build their analysis tools and more customized applications.

IV. Evaluation

We set up our blockchain network on three virtual servers using a cloud service. Each of them is equipped with two cores of an Intel Xeon Gold 6248R CPU @ 3.0 GHz, 8GB memory, and 64GB storage with Redhat Linux OS. Our servers are set up on the same nodes to reduce network latency. Our smartphone apps are developed in Android Studio and Apple Xcode, supporting the latest Android 12 and iOS 15

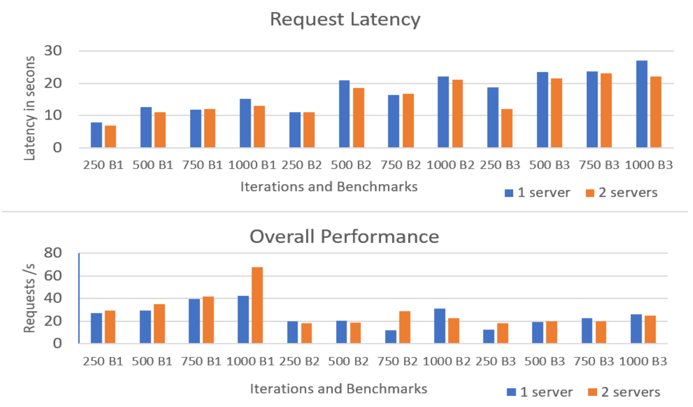

Fig. 7. Latency and Performance

respectively, and installed on a variety of models such as Samsung, Google Pixel, and iPhone.

To evaluate our proposed design, we did extensive experiments revealing the overall performance of the whole system. We created concurrent requests using Postman and flooded them into the server to mimic the concurrent requests from users. As Figure 7 shows, we have three groups of benchmarks: B1 is the user's contact update, which sends a transaction with the contact information to the server and the transaction is recorded by the blockchain. B2 is the same user's update as B1, plus a generation of the graph by pulling blockchain data from the server. B3 is the back-trace function, which sends a positive result to the server and triggers the back-trace analysis algorithm. As the x-axis labels indicate, we send 250, 500, 750, and 1000 copies of each benchmark concurrently reflecting the different scales of users. Meanwhile, we repeat the same benchmark on a 1-server setup and a balanced 2-server setup.

We measured two metrics for the performance: the latency and the performance. The latency defines as the average time between a request being fired and a response being gotten, measured in seconds. In the top half of Figure 7, it was expected that the latency would go up as the requests increase, but using a second server will always reduce the response latency. The performance, shown at the bottom part is the average number of requests processed per second (r/s). B1 is much faster since it does not trigger the database writing and the maximum speed is 67 r/s on two servers. On the right, pulling up the contact from the blockchain takes more time and dragged the number down to 20 r/s on one server and 22 r/s on two servers.

V. Conclusion

In this paper, we proposed GeauxTrace, a scalable privacy-protecting contact tracing platform. It detects physical contact using the Bluetooth module in the app and stores the proof of contact anonymously on the private blockchain. Then through the server, we can generate a global graph of the contact graph and notify the multi-level users at risk. Our platform can also connect with other upstream detecting apps and downstream analysis tools via APIs. The evaluation shows the feasibility and performance of the platform in real-world scenarios.

VI. Acknowledgement

This work is supported in part by an NIH grant 1U01AA029348-01. Anonymous referees provide helpful comments.

References

[1] https://www.google.com/covid19/exposurenotifications/.
[2] https://www.hyperledger.org/.
[3] E. Bandara, X. Liang, P. Foytik, S. Shetty, C. Hall, D. Bowden, N. Ranasinghe, and K. De Zoysa, "A blockchain empowered and privacy preserving digital contact tracing platform," *Information Processing & Management*, vol. 58, no. 4, p. 102572, 2021.
[4] 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.
[5] M. Cunche, A. Boutet, C. Castelluccia, C. Lauradoux, and V. Roca, "On using bluetooth-low-energy for contact tracing," Ph.D. dissertation, Inria Grenoble Rhône-Alpes; INSA de Lyon, 2020.
[6] 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.
[7] J. Han, J. Pei, and H. Tong, *Data mining: concepts and techniques*. Morgan kaufmann, 2022.
[8] W. H. Organization *et al.*, "Ethical considerations to guide the use of digital proximity tracking technologies for covid-19 contact tracing," 2020.
[9] L. Setti, F. Passarini, G. De Gennaro, P. Barbieri, M. G. Perrone, M. Borelli, J. Palmisani, A. Di Gilio, P. Piscitelli, and A. Miani, "Airborne transmission route of covid-19: Why 2 meters/6 feet of inter-personal distance could not be enough," p. 2932, 2020.
[10] B. Xu, D. Luthra, Z. Cole, and N. Blakely, "Eos: An architectural, performance, and economic analysis," 2018.
[11] H. Xu, L. Zhang, O. Onireti, Y. Fang, W. J. Buchanan, and M. A. Imran, "Beeptrace: blockchain-enabled privacy-preserving contact tracing for covid-19 pandemic and beyond," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3915–3929, 2020.