ABAFOR: A Blockchain-based Privacy-Preserving Architecture for Efficient Contact Tracing and GIS Analysis

Syed Ibtisam Tauhidi, Abdullahi Abubakar, Abdulhakeem Ishola, Alhaji Idi Babate, Zayyanu Umar, Musa Abubakar Alkali Tanko, Ahmad Maccido Abdulkadir, Umar Audu Mustapha, Muhammad Haruna Rasheed and Hashiru Isiaka Muhammad

***Abstract*—The Coronavirus disease 2019 has manifested into a global pandemic spreading into almost all the countries and territories in the world. Contact tracing, followed by testing and isolation, have been identified as important tools in containing the proliferation of the disease. Because of manual contact tracing limitations, smartphone apps for digital contact tracing have been deployed by authorities in multiple countries. However, many of these apps have faced criticism because of interoperability, privacy and security issues. This paper proposes an open architecture based on blockchain technology that addresses some of these criticisms. It enables interoperability through a publicly-readable consortium blockchain database, preserving privacy by distributing users' partial identities among multiple decentralised authorities and providing security through digital signature and asymmetric key encryption. In addition, the architecture provides a fast proof-of-authority based consensus algorithm using reputation as a stake to add and validate blocks on the blockchain efficiently and a swift contact tracing algorithm using Spatio-temporal and key-valued databases.**

***Index Terms*—Blockchain, Contact tracing, Covid-19, Geographic Information System (GIS), Pandemic, Privacy preservation.**

I. INTRODUCTION

As of this writing, the Coronavirus disease 2019, also called COVID-19, has already infected more than 500 million people worldwide, causing an excess of 6.2 million deaths [1], making it one of the deadliest pandemics in history. Without the availability of a vaccine, it is, therefore, of utmost importance that the spread of the disease be contained.

The effectiveness of contact tracing in containing contagious diseases has already been proven in the elimination of smallpox [2]. It is asserted [3] that contact tracing, increased testing, and self-isolation should be conducted despite their high short-term cost to avoid the higher long-term economic and social cost of social distancing.

Traditionally, contact tracing has relied on manually interviewing patients and their contacts, but this process is limited by the memory and attention of the interviewees. These limitations are even more pronounced in finding secondary and tertiary contacts.

Moreover, because of the disease's highly infectious and low virulent nature, manual contact tracing is deemed insufficient to contain the spread of COVID-19. However, mathematical simulations have demonstrated that a large proportion of the population using a digital contact tracing application (hereafter called *app*) would be efficient in controlling the pandemic [4].

Predictably, Location-based Services (LBS) [5; 6] for digital contact tracing using smartphone apps have gained prominence in recent times. However, it should be noted that similar ideas have existed since 2007 [7; 8]. Examples of such apps, in the context of COVID-19, are Aarogya Setu [9] and TraceTogether [10] deployed by the governments of India and Singapore, respectively.

Although the usage of digital contact tracing apps have resulted in relative success in containing the spread of COVID-19, these apps are not without their share of criticisms. Most of these criticisms can be categorised as follows.

1) **Closed ecosystems:** Each app is deployed by a different service provider with no interoperability with other apps. This segments a user's data when he or she moves from a region being served by a particular provider to a different region.
2) **Vulnerability to DoS attacks:** A *Denial-of-Service* attack on an app's centralised server could make the service of the app unavailable.
3) **Mutability of data:** A malicious user with write-access to the centralised database can modify the data to tamper

ISSN: 2736-5751
DOI: http://dx.doi.org/10.24018/ejece.2022.6.2.434
Published on April 30, 2022.

S. I. Tauhidi, School of Electronics, Electrical Engineering and Computer Science, Queen's University, UK. (Correspondinge-mail: stauhidi01.aabubakar02@qub.ac.uk)
A. Abubakar, Department of Computer Science, Waziri Umaru Federal Polytechnic, Nigeria. (e-mail: abafo22@gmail.com)
A. Ishola, Department of Computer Science, Waziri Umaru Federal Polytechnic, Nigeria. (e-mail: abduljedo@gmail.com)
A. I. Babate, Department of Computer Science, Federal College of Education (Tech.), Nigeria. (e-mail: ababate2002@gmail.com)
Z. Umar, Department of Computer Science, Waziri Umaru Federal Polytechnic, Nigeria. (e-mail: suwaizay@gmail.com)
M. A. Alkali Tanko, Department of Computer Science, Waziri Umaru Federal Polytechnic, Nigeria. (e-mail: musaibnabubakar@gmail.com)
A. M. Abdulkadir, Department of Computer Science, Waziri Umaru Federal Polytechnic, Nigeria. (e-mail: amprehmaccido@gmail.com)
U. A. Mustapha, Department of Computer Science, Waziri Umaru Federal Polytechnic, Nigeria. (e-mail: umarumustapha955@gmail.com)
M. H. Rasheed, Department of Computer Science, Waziri Umaru Federal Polytechnic, Nigeria. (e-mail: hrmuhammad87@gmail.com)
H. I. Muhammad, Department of Computer Science, Abubakar Tafawa Balewa University, Nigeria. (e-mail: hashiruisiaka@gmail.com)

with the contact tracing process, creating chaos.

4) **Lack of privacy:** Unencrypted data associated with a user's identity can be misused to track the user. On the other hand, encrypting a user's data prevents the analysis of trends and patterns in the spread of the disease.
5) **Non-transparency:** Intentional or unintentional errors in the contact tracing process cannot be scrutinised because of the proprietary data being owned privately by the centralised authorities. i.e. app providers.
6) **Centralised trust:** The users need to trust the app provider's capability of providing security and privacy[11], and preventing data misuse.
7) **Restrictions of unimodal data collection:** A unimodal app using only GPS data would have inaccuracies because of the inherent inaccuracies of the GPS sensors. For example, all people in a high-rise building would have the same latitude and longitude, despite being distant from one another. This results in a high false-positive error rate. On the other hand, GIS analysis of trends and patterns in the spread of the disease cannot be done on data collected by a unimodal app that exchanges location-free Bluetooth tokens.

This paper proposes an open architecture for a smartphone app that trades off total privacy in favour of partial privacy to enable transparent contact tracing and GIS (Geographic Information System) data analysis. The users of a smartphone app built on top of the proposed architecture upload encrypted location and infection status information using a round-robin algorithm to multiple *miners*, called *gateways*, of a blockchain.

The gateways, which can identify the users with their IP addresses and public keys, decrypt and anonymise the user's uploaded data before adding it to the blockchain. The gateways maintain records allowing them to identify probable contacts after contact tracing, and to notify said contacts.

The architecture exposes each user's instantaneous location to a particular authority (i.e., gateway) no more than once every 12 hours. Despite access to partial identities of the users, the gateways' abilities to track a user is restricted by this limit on the gateways' sampling rate of the users' data.

The architecture includes a publicly-readable *consortium* blockchain that can be used by multiple app providers to deploy a myriad of contact tracing apps built on top of the architecture, thus enabling interoperability among all such apps. Data anonymisation preserves privacy on the blockchain.

The architecture provides the following features.

1) The decentralisation, immutability, and transparency of data are provided by the use of a consortium blockchain.
2) The consortium blockchain eliminates the necessity of the users' trust in centralised authorities. The architecture works as long as the majority of the gateways in the consortium are honest. Dishonest gateways are automatically purged from the network.
3) The geographic and infection status of the users are stored in the publicly-readable blockchain. This enables transparent contact tracing and GIS analysis. Privacy is preserved by disassociating the users' identities from their data in the blockchain.
4) The users add data to the blockchain through intermediate gateways. A gateway's ability to track a user is diluted by limiting the rate at which it receives data from the user.
5) The Spatio-temporal data on the blockchain is used to identify probable contacts who were in the vicinity of an infected patient. The users' computation and network resource expenses are reduced by performing this initial contact tracing at the gateways, which are expected to be run on a parallel, high-performance infrastructure. Once the initial contact tracing process is complete, the gateways notify the probable contacts to carry out the second phase of the contact tracing process.
6) A multimodal data acquisition method using both GPS sensors and Bluetooth tokens eliminates false negatives. The probable contacts self-assess the risk of being infected using the Bluetooth Received Signal Strength Indicator (RSSI) to estimate proximity to infected or at-risk users.
7) The two-step contact tracing process finds the set of primary contacts first. Iterative application of the process finds the secondary contacts, tertiary contacts, and so on.

The rest of the paper is divided as follows. Section II reviews related works in privacy-preserving contact tracing; Section III describes the motivation behind the development of the proposed architecture; Section IV gives an overview of the system design; Section V describes the detailed design of the architecture; Section VII discusses the properties of the proposed architecture, and Section VIII concludes.

II. Related Work

In the context of contact tracing, [12] identifies three notions of privacy - (i) privacy from snoopers, (ii) privacy from contacts, and (iii) privacy from the authorities. The *privacy from snoopers* notion prevents the data being transmitted by an infected or uninfected user to a trusted or trustless receiver from being intercepted by any other entity. The *privacy from contacts* requirement ensures that primary, secondary or tertiary contacts of an infected user are to be determined without revealing the identity of the infected user. Finally, the *privacy from authorities* notion guarantees that the agencies or organisations responsible for deploying the contact tracing system are unable to track the location of an infected or uninfected user. Moreover, the paper notes that any architecture would need to reveal *some* data to achieve contact tracing.

Further, the possibility of a linkage attack being ingrained in contact tracing is illustrated. For example, if a contact was near only one other person in the previous two weeks, then the infection status of the other person can be inferred.

The South Korean model of contact tracing during COVID-19 is discussed in [13], wherein the government publishes in the public domain the diagnosed patients' pre-infection movements, tracked through GPS phone tracking, credit card records, surveillance video, and

interviews. Both *privacy from contacts* and *privacy from the authorities* of the patient is traded off to enhance the privacy of individuals trying to determine their exposure.

A widely adopted specification, named Exposure Notification System, for privacy-preserving contact tracing, based on Decentralised Privacy-Preserving Proximity Tracing (DP3T) [14] system, is proposed by Apple and Google in [**?**]. The users generate pseudo-random keys once every 10 to 15 minutes and frequently broadcast tokens with the keys through Bluetooth. Additionally, the users store the tokens received from other users. When a patient is diagnosed with COVID-19, he or she uploads the keys generated by them to a trusted server, which is then broadcast to all users of the system. The users match the keys received from the server with the keys received through Bluetooth tokens to perform contact tracing.

Thus, privacy is enabled by hiding users' identities, although infected users are required to be validated by authorised entities before uploading data to the centralised server. Although this system uses decentralised contact tracing in users' smartphones, a central reporting server is used to store the keys of infected users. This unimodal solution does not use location data, thereby preventing any analysis of the trends and patterns in the spread of the disease. Moreover, this enables the tracing of only primary contacts. There is no mechanism to enable the tracing of secondary or tertiary contacts. Despite its limitations, this system is adopted for use by authorities in Austria, Belgium, Croatia, Germany, Ireland, Italy, the Netherlands, and Switzerland, among others.

A privacy-preserving peer-to-peer smartphone app for contact tracing is developed in [15]. The app maintains a data structure called the *transmission graph*. The graph's nodes, called *contact point*, are created when users in proximity manually scan a QR code. When a user moves from one *contact point* to a different *contact point*, an edge, called a *transmission vector*, connects the two nodes.

A centralized server enables infected users to confirm their positive diagnosis. For any target *contact point*, the *transmission graph* is used to determine *possible transmission paths*, i.e. one or multiple paths of contacts from *contact point*s with infected users to the target *contact point*.

This approach preserves a user's privacy by not storing any location or identity information in the *transmission graph*. However, it is predicted that user fatigue might result from the users repeatedly adding themselves manually to the various contact points, thus resulting in decreased adoption. Moreover, each user maintains a local subgraph of the global *transmission graph*. Thus, the analysis of global trends and patterns in the graph is not feasible.

Another centralised contact tracing architecture, EPIC, is designed in [16], wherein a server maintains unencrypted data of infected users and encrypted data of other users. The privacy of infected users is traded off to preserve the privacy of other users. Moreover, this architecture uses wi-fi signal strength received from at least three access points to calculate matching scores indicating the proximity of a user to an infected user, requiring the deployment of such a wi-fi infrastructure.

A decentralized contact tracing architecture is given in [17]. Users generate temporary IDs, broadcast them using Bluetooth, and store the IDs received from other users' broadcasts. A globally-accessible distributed hash table (DHT) using each user's IDs as keys is maintained. On being diagnosed as infected, a user alerts his or her primary contacts by updating the value at the DHT pointed to by the user's stored list of received IDs. Centralised authorities restrict malicious data mutation using privacy-preserving blind signatures to authorise only infected users to update the DHT.

The blockchain technology enables a decentralized and distributed cryptographic linked list of immutable records. It was brought to prominence by the introduction of the cryptocurrency Bitcoin [18], and has been adopted for general-purpose computation using *smart contracts* and *dapps* by frameworks like Ethereum [19] and Hyperledger [20]. Blockchains have been adapted for use in numerous applications [21; 22; 23; 24; 25; 26], including GIS and contact tracing.

The use of blockchain technologies in Public Participation Geographic Information System (PPGIS) to improve local public involvement and awareness of geographic decision-making is demonstrated in [27]. The use of blockchain improves accountability, openness, and transparency in the PPGIS application by making the data available publicly for scrutiny and audit and making the application code public through a smart contract on the Ethereum platform. The utility of such a PPGIS architecture is demonstrated by developing a decentralised application that enables users to vote for the selection of sites for urban utilities. The authors point out the limitations of Solidity and Ethereum in storing geographical data using spatial data structures and the higher cost of storing a large amount of data on the blockchain. These limitations make Ethereum inefficient for use in GIS applications using high volume of data.

Another application of blockchain technologies, in the context of GIS, is proposed in [28]. The architecture uses RFID along with GPS-based G.I.S. applications to track the transportation of agri-products from farms to plants, warehouses, and eventually the sales markets. The deployment of production, warehouse management, sales and transportation data on immutable and public blockchains improve tracking and traceability of agri-products, enhance their credibility, and fight fake products.

In Haiti, after the loss of centralised land registry records in the disastrous earthquake in 2010, there were difficulties in identifying owners of several plots of land. Moreover, disputes arose over several plots of land being claimed by multiple individuals. These problems could have been avoided by the use of immutable, distributed land records. To prevent similar problems, blockchain-based land registry records are being deployed in Brazil [29], Honduras [30], India [31], and Sweden [32].

A solution to address the issue of the dispute of boundary of land plots is further proposed in [33]. Multiple cadastre

surveys of the same real estate might be undertaken at multiple times, thereby enabling the possibility of inconsistency and non-interoperability of the data. The proposed system uses a multisig contract and GIS/CAD modules to provide consistency in the data. Land boundaries are measured with GIS applications and further segmented using CAD applications. These survey records are stored on a blockchain. In addition, the system cross-checks records of previous cadastre surveys on the blockchain to prevent inconsistency in record-keeping.

Blockchain technologies have found several applications in the context of the ongoing Coronavirus pandemic. The utility of blockchain in issuing *immunity passport* or *immunity certificates* has been explored in [34]. It is suggested that individuals who have been infected with and are immune to COVID-19 be issued such certificates and be allowed to resume normal daily activities. The paper also discusses the idea of a smart contract in the blockchain that assigns an ID to each user's contact with another user, enabling contact tracing by authorities. Privacy preservation has not been addressed in the paper.

A privacy-preserving blockchain-based solution using the zero-knowledge proof protocol and the key escrow mechanism has been outlined in [35]. The solution uses short-range communication enabled IoT devices as *honesty witnesses* that validate contacts between users on the blockchain. Thus, such IoT devices must be deployed wherever contacts between individuals might occur.

Another blockchain-based architecture, BeepTrace, for privacy-preserving contact tracing is given in [36]. Two blockchains, one (*tracing blockchain*) to store encrypted spatial data and another (*notification blockchain*) to alert contacts, are maintained. BeepTrace requires its users to trust a centralized *geo-solver* to decrypt data on the *tracing blockchain* and perform contact tracing. Thus, *privacy from the authorities* is not preserved in this architecture.

III. Motivation

The various smartphone app-based digital contact tracing techniques reviewed in the literature, including those based on blockchain technology, have at least one of two major drawbacks. Either a centralised authority is trusted to restrict data mutation (to perform contact tracing), or the user undertakes self contact tracing, hiding all information from all other actors in the system.

Blockchain technology is inherently trustless. Therefore, imposing trust on one centralised authority undermines the usage of a blockchain. Moreover, a malicious user gaining administrative access to any trusted centralised authority could compromise a user's privacy.

On the other hand, although hiding the users' location and infection status information from other actors in the system preserves a user's privacy, such an approach disables the ability to analyse the data to discover trends and patterns in the spread of the disease.

This paper intends to propose an architecture with complete *trustlessness* and without any role for centralised authorities. The architecture should allow for the development of interoperable smartphone apps for contact tracing while providing security and preserving all three notions of privacy. Furthermore, despite the stringent security and privacy requirements, the users' anonymised data should be made available in the public domain for transparent contact tracing and GIS data analysis.

IV. System Design

A. The different actors in the system

There are two mutually exclusive sets of actors in the system - the set of the various smartphone apps' end-users, Ω, and the set of gateways, Φ. Each actor maintains a pair of asymmetric keys. The notations $\psi_i \in \Omega$ and $\xi_j \in \Phi$ refers to arbitrary i-th end-user and j-th gateway, respectively.

TABLE I: Notations

Symbol	Meaning
Ω	*Set of end-users*
Φ	*Set of gateways*
ψ_i	*An arbitrary end-user.* $\psi_i \in \Omega$
ξ_j	*An arbitrary gateway.* $\xi_i \in \Phi$
ξ_j^{HM}	*The honesty-metric of* ξ_i
Φ_κ^i	*Set of known gateways of* ψ_i
Φ_α^t	*Set of active gateways at time* t
ψ_i^{PUK}	*Public key of* ψ_i
ψ_i^{IP}	*Current IP address of* ψ_i
ψ_i^{PRN}	*Latest pseudorandom number generated by* ψ_i
ψ_i^F	*Flag of* ψ_i*'s latest upload-data message*
ψ_i^{LAT}	*Latitude of* ψ_i*'s latest upload-data message*
ψ_i^{LNG}	*Longitude of* ψ_i*'s latest upload-data message*
ψ_i^H	*Hash of* ψ_i*'s latest upload-data message*
ψ_i^T	*Type bits in* ψ_i^F
ψ_i^S	*Status bits in* ψ_i^F
ψ_i^{CH}	*Changed bit in* ψ_i^F
ψ_i^{TS}	*Timestamp of* ξ_j *receiving* ψ_i*'s data packet*
P	*Set of infected users.* $P \subseteq \Omega$
C	*Set of probable contacts.* $P \subseteq \Omega$
τ	*Rate at which* ψ_i *uploads upload-data message*

In the proposed architecture, gateways play the role *miners* traditionally play in blockchains like Ethereum. Gateways are being called to reflect their added responsibility of interfacing the end-users in Ω to the blockchain. Gateways receive data packets from the end-users, add the data as *transactions* into blocks, append the blocks to the blockchain, perform contact tracing, and notify probable contacts.

The subset of Φ known to ψ_i even before ψ_i's first use of the app is denoted by Φ_κ^i. Φ_κ^i is available in the public domain and mirrored in servers across multiple geographies. ψ_i's app downloads Φ_κ^i from any of the mirrors before starting to use the app.

TABLE II: Types of messages exchanged between different actors.

Message Type	TYPE BITS	Sender	Receiver	Response
query-gateway	000	User	Gateway	*active-gateway*
request-to-join	001	User	Gateway	Binary
upload-data	111	User	Gateway	None
active-gateway	101	Gateway	User	None
purge-gateway	010	Gateway	User	None
notify-contact	011	Gateway	User	None
keep-tracing	110	Gateway	Gateway	None
heartbeat	100	Gateway	Gateway	None

Gateways are not expected to be always active. Apart from common issues like power and network outages, events like DoS attacks might make one or more gateways unavailable. Thus, the set Φ is dynamic in time. The set of all active gateways at time t is denoted by Φ_α^t.

Two subsets of Ω are identified in the context of contact tracing - the set of infected patients, $P \subseteq \Omega$, and the set of contacts, $C \subseteq \Omega$. The set P is used to find the primary contacts, which are in turn used to find the secondary contacts, and so on, iteratively forming the set C.

Table I lists the notations used in the rest of the paper.

B. Messages exchanged between actors

The different actors in the network communicate with each other by exchanging various types of messages (see Table II) using the TCP/IP protocol stack. In IPv6, each data packet has a 62 bytes header, which includes the 60 bytes TCP/IPv6 header and a 2 bytes `VERSION` field set to `0×0000`. In IPv4, the header size reduces to 42 bytes.

The `VERSION` would change in the future if a *hard fork* makes the architecture backward-incompatible or if additional infectious diseases would require contact tracing. Following the header, each message type has a 1 byte `FLAG` field (see Fig. 1). The `TYPE BITS` in `FLAG` indicate the type of the message, while the `CHANGED BIT` and the `STATUS BITS` are used exclusively for contact tracing and analysis. In all message types, except *upload-data*, the `CHANGED BIT` and the `STATUS BITS` are set to `0`s.

After downloading Φ_κ^i from a mirror, ψ_i queries for Φ_α^t using *query-gateway* messages. The *query-gateway* messages set the `FLAG` to `0×00` and contain no payload.

On receiving a *query-gateway* message, a gateway responses with an *active-gateway* message containing Φ_α^t and the public keys of the gateways in Φ_α^t, with the `FLAG` set to `0×A0`. The `FLAG` is followed by a 4 byte unsigned integer containing $|\Phi_\alpha^t|$. A sequence of 48 byte blocks, each containing a 16 byte IP addresses and a 32 byte public key of a gateway in Φ_α^t, is appended to the data packet.

Finally, ψ_i completes the registration process by sending *request-to-join* messages to the gateways in Φ_α^t. The *request-to-join* message sets the `FLAG` to `0×30`. A 32 byte public key of the user, ψ_i^{PUK}, and a 1 byte signed and encrypted hash of the public key is appended to the payload.

The user, ψ_i, uploads his or her current location data to different gateways per τ units of time using the *upload-data* message. The payload of the message, illustrated in Fig 2, contains four fields: 4 bytes each of user's current latitude (ψ_i^{LAT}) and longitude (ψ_i^{LNG}), a 32 bytes pseudorandom number (ψ_i^{PRN}), and a 1 byte hash (ψ_i^H). Methods to set the values of the various fields in the *upload-data* message are further discussed in Section V-C.

When ξ_j adds a block to the blockchain, the consortium validates the block and increments or decrements ξ_j's *honesty-metric*, ξ_j^{HM}, depending, respectively, on the success or failure of the validation. If ξ_j^{HM} falls below 0, ξ_j is assumed to be dishonest and is removed from the consortium. All other gateways in the consortium inform their registered users about this event using a *purge-gateway* message. The *purge-gateway* message sets the `FLAG` to `0×40`, and contains the 16 byte IP address of ξ_j.

Suppose ψ_i informs ξ_j about being diagnosed with or exposed to the disease. In that case, ξ_j starts the initial contact tracing process to find all probable contacts of ψ_i known to it and sends *notify-contact* messages alerting those probable contacts. The list of pseudo-random numbers (hereafter called PRNs) generated by ψ_i is sent with the message. The *notify-contact* message sets the `FLAG` to `0×60`, and contains a 4 byte unsigned integer indicating the count of the PRNs and a 1 byte copy of ψ_i's most recent *upload-data* message's `FLAG`, followed by blocks of 32-byte PRNs added sequentially after the count.

Because of user-identifiable data being distributed among the gateways, ξ_j can identify and alert only a subset of all potential contacts of ψ_i. ξ_j notifies all other gateways to proceed with and complete the contact tracing process using a *keep-tracing* message. The *keep-tracing* message sets the `FLAG` to `0×C0`, and contains two 4 byte unsigned integers - the first integer is set to the count of ψ_i's PRNs, and the second one is set to the count of the PRNs of ψ_i's potential contacts. Following the integers is the sequences of 32 byte PRNs generated by ψ_i. Then, another sequence of 32 byte PRNs of ψ_i's potential contacts is appended.

The gateways broadcast *heartbeat* messages once per τ units of time to all active gateways in the network. The *heartbeat* message has the `FLAG` set to `0×80`, and contains a 1 byte signed and encrypted hash of the public key of the sender.

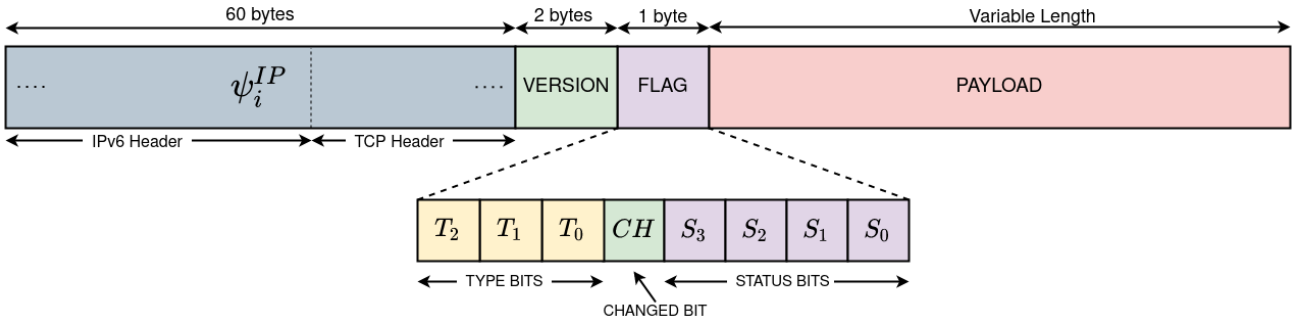

Fig. 1: TCP/IPv6 data packet format

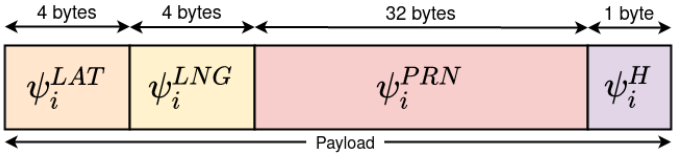

Fig. 2: Payload of an *upload-data* message

This is used by gateways to maintain the updated list of other active gateways.

C. Encryption, hashing and decryption of messages

Both *query-gateway* messages send by ψ_i to gateways in Φ_κ^i and *active-gateway* messages responded by the gateways in Φ_κ^i to ψ_i are unencrypted. The `FLAG` and PUK fields of *request-to-join* messages are unencrypted, but the hash is signed and encrypted by ψ_i before being sent. This enables ξ_j to verify the integrity of the exchanged keys.

The *query-gateway*, *active-gateway*, and *request-to-join* messages do not contain any sensitive data that could compromise a user's privacy. This is in contrast to the *upload-data* messages, where all the fields, including the `FLAG`, contain sensitive data.

In an *upload-data* message, the hash is computed on the unencrypted `FLAG`, the user's current latitude and longitude, and the user's most recently generated PRN encoded in the message. Then, the five fields, including the hash, are signed and encrypted by ψ_i before being sent to ξ_j.

When ξ_j receives a data packet from ψ_i's registered IP address ψ_i^{IP}, it does not know if the `TYPE BITS` corresponds to an encrypted *upload-data* message from ψ_i, or an unencrypted *query-gateway* or *request-to-join* message from another user, ψ_j, who is assigned the IP address previously assigned to ψ_i.

The *upload-data* messages form the bulk of the messages sent by a user to the gateways. Hence, ξ_j assumes any received message from ψ_i^{IP} to be an *upload-data* message from ψ_i, and decrypts the message using ψ_i^{PUK} and its own private key. Then, ξ_j validates the hash of the message. If the hash validates successfully, ξ_j checks the `TYPE BITS` to confirm that the message is indeed of type *upload-data*.

The hash validation fails if the message is, instead, of type *query-gateway* or *request-to-join* from ψ_j. It should be noted that failure in hash validation because of data corruption is unlikely with TCP/IP because of *checksum* providing data integrity. In case of failure in hash validation, ξ_j checks the `TYPE BITS` without decryption to recognise the type of message.

Apart from *active-gateway* messages, ξ_j may send *purge-gateway* and *notify-contact* messages to ψ_j. The `FLAG` of neither of the two message types contain any sensitive data. Hence, the `FLAG` of all message types sent by ξ_j to ψ_j are unencrypted. Thus, ψ_j can check the `TYPE BITS` of any message from ξ_j without any decryption to determine the message type.

Moreover, the *purge-gateway* message with a dishonest gateway's IP address is broadcast to all users. Hence, encrypting the data in the message would not hide any information from a snooper. Therefore, ψ_j transmits the entire *purge-gateway* message without any encryption.

On the other hand, the *notify-contact* message contains a diagnosed patient's or a contact's `FLAG` and the set of PRNs generated by the patient or contact. Although the information in a *notify-contact* message is anonymised and lacks any user-identifiable data, these are signed and encrypted to hide them from snoopers.

None of the three message types sent by ξ_j to ψ_i contains any sensitive data in the `FLAG`. Hence, the `FLAG` is unencrypted in all the three message types. Thus, ψ_i can directly look up the `TYPE BITS` in ξ_j's messages to determine the message type.

In the hashing, signing, and encryption process described above, the Pearson hashing function is used to generate the 8-bit hash. The RSA algorithm is used for digital signature and encryption using asymmetric keys whenever required.

The Pearson hashing algorithm is cryptographically insecure, but it is sufficient for the key validation and message type checking applications in the proposed architecture, wherein the hashing function is not used for any cryptographic application. Further, the security of the RSA algorithm relies on the difficulty of the integer factorisation problem, which can be compromised on quantum computers

using Shor's algorithm. Therefore, the proposed algorithm would require a *hard fork* with quantum cryptographic algorithms when such quantum computers would become available for public use. The `VERSION` field in the header should be incremented to accommodate such a *hard fork*.

V. Methodology

The following subsections describe the various stages in the working of the proposed architecture, along with data structures and algorithms that should be implemented by app providers and gateways for compatibility with the architecture.

A. A user obtains the set of active gateways

A user, ψ_i, registers with the gateways before starting to upload data. ψ_i begins the registration process at time t by sending *query-gateway* messages to all known gateways, Φ_κ^i. The gateways in Φ_κ^i responds with the set of active gateways, Φ_α^t and the public keys of the gateways in Φ_α^t using *active-gateway* messages.

The unavailable gateways in Φ_κ^i fail to respond and are ignored. In case of discrepancies in versions of Φ_α^t received from multiple gateways, ψ_i uses majority voting to choose the version of Φ_α^t received from most gateways. Finally, Φ_α^t is permuted into an ordered set. The order of Φ_α^t is invariant until the user's re-registration.

B. User registers with all the active gateways

ψ_i broadcasts *request-to-join* messages, along with ψ_i's public key, to all gateways in the ordered set Φ_α^t. The gateways respond with a binary value indicating success or failure of registration. In case of failure, ψ_i attempts re-registration by resending *query-gateway* and *request-to-join* messages sequentially.

The user, ψ_i, also re-registers whenever any of the gateways in Φ_α^t becomes unavailable. A TCP error can identify the unavailability of a gateway during ψ_i's attempt to send an *upload-data* message. After re-registration at time t', the updated set of active gateways, $\Phi_\alpha^{t'}$, would have eliminated all inactive gateways present in Φ_α^t, and added all gateways which turned active between time t and t'.

C. User uploads data to gateways

After completing the registration process, ψ_i uploads data packets with location and infection status information to the gateways in the ordered set Φ_α^t using a round-robin algorithm. The data is uploaded with an *upload-data* message, once per τ units of time.

The value of τ dynamically adjusts, based on Φ_α^t, such that a gateway, ξ_j, receives *upload-data* message from ψ_i no more than twice per day.

The following subsections discuss the four fields in an *upload-data* message sent by ψ_i to ξ_j.

1) Flag (ψ_i^F): The position of each bit in the flag is shown in Fig. 1. The three `TYPE BITS`, $\psi_i^T, (T_2 - T_0)$ are set to `111`. The four `STATUS BITS`, $\psi_i^S, (S_3 - S_0)$, indicate the probability of the user being infected, wherein value `0×0` indicates that the user is not at risk, and `0×F` indicates that the user is diagnosed with the disease. The `CHANGED BIT`, ψ_i^{CH}, being set to 1 indicates that ψ_i^S has changed since ψ_i's the last *upload-data* to ξ_j. The algorithm to compute ψ_i^S and ψ_i^{CH} is specified in Section V-J.

2) Latitude (ψ_i^{LAT}) & Longitude (ψ_i^{LNG}): The floating-point values of the *decimal degrees* representation, (LAT^{DD}, LNG^{DD}), of ψ_i's current location are transformed into 4 byte integers (ψ_i^{LAT}, ψ_i^{LNG}) using Eq. 1. This integer representation enables efficient storage and computation while preserving precision with an error of $\pm$111.32mm. This is sufficient accuracy for contact tracing purposes.

$$(\psi_i^{LAT}, \psi_i^{LNG}) = round(LAT^{DD} \times 10^6),\ round(LNG^{DD} \times 10^6) \tag{1}$$

Although 28 and 29 bits are required to encode the ψ_i^{LAT} and ψ_i^{LNG} respectively, 32 bits are allocated to each of the two fields to keep them byte-aligned. Despite the tradeoff, assuming IPv6 protocol and $\tau = 1$ minute, around 51MB bandwidth is utilised by the *upload-data* messages per user per year.

3) Pseudorandom Number (ψ_i^{PRN}): In the initiatory usage of the contact tracing smartphone app, ψ_i seeds the Mersenne Twister PRN generator algorithm with the 32-byte hash of ψ_i's private key. The state of the PRN generator is stored in persistent memory. Whenever the app is closed and restarted (e.g., after rebooting the smartphone), the state is reloaded from persistent memory.

As already discussed, ψ_i uploads *upload-data* messages using a round-robin algorithm on the ordered set Φ_α^t, i.e., each subsequent data packet is uploaded to a different gateway exhausting all gateways in Φ_α^t per iteration. ψ_i sequentially generates a different PRN for each *upload-data* message.

4) Hash (ψ_i^H): Hash is calculated on the previous four fields using Pearson hashing. The hash serves two purposes - first, it ensures the integrity of the received data packets. However, failure to validate a hash mainly indicates that the user of an IP address might have changed. When ξ_j receives a data packet from an IP address, ψ_i^{IP}, it assumes the data packet to be of type *upload-data* sent by ψ_i. This scenario forms the bulk of the traffic in the network. ξ_j decrypts the message using ψ_i^{PUK}. If the user of the IP address changes, the decryption results in failure in hash validation.

D. User broadcasts data through Bluetooth

Apart from sending *upload-data* messages to the active gateways, ψ_i broadcasts the last generated ψ_i^{PRN} through

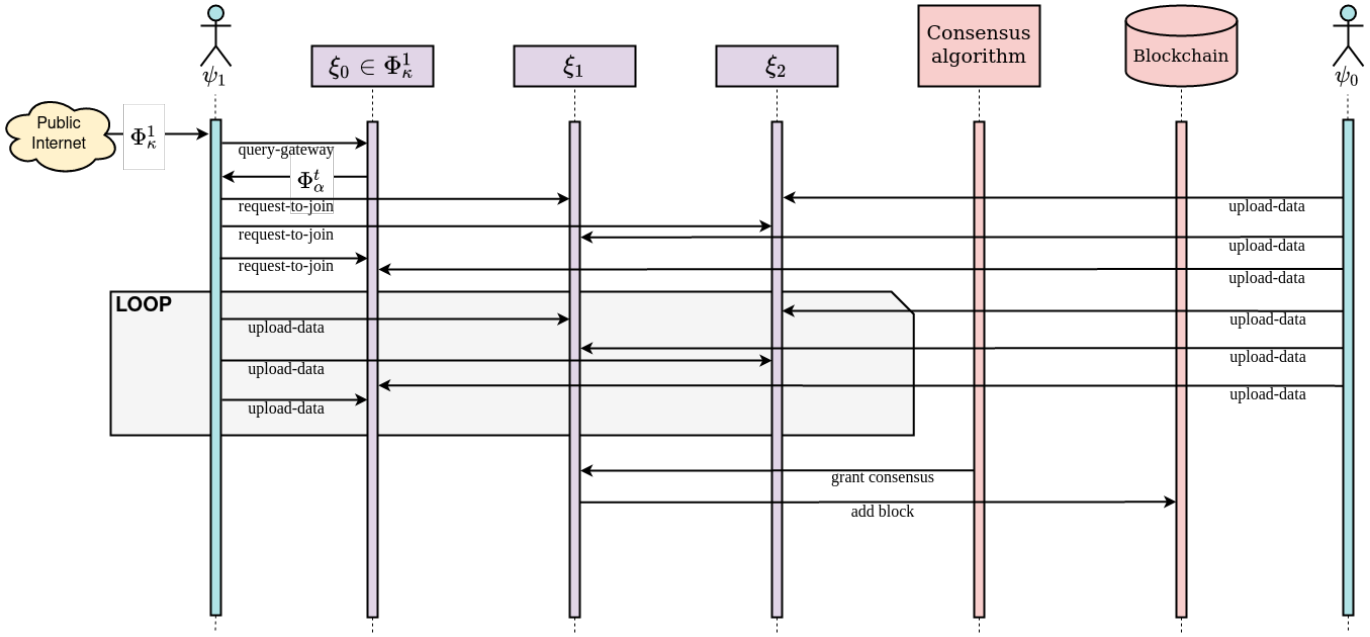

Fig. 3: User initialization to the addition of a block to the blockchain

Bluetooth 16 times per τ units of time, and receives broadcasts from other users at the same frequency.

ψ_i retrieves ψ_j^{PRN} and Received Signal Strength Indication (ψ_j^{RSSI}) from the received broadcast tokens from another user, ψ_j, to determine ψ_i's *proximity* to ψ_j. The value of *proximity* is scaled to fit an unsigned half byte. A hash table, BT, using ψ_j^{PRN} as key and $proximity$ as value is persistently stored by ψ_i.

It should be noted that *proximity* should describe the risk of transmission of the disease associated with a contact, and not the actual distance. For example, two contacts who were momentarily close to one another would have a lower risk of transmitting the disease among themselves compared to two contacts who were in close vicinity for an extended duration. ψ_i broadcasts Bluetooth tokens 16 times per τ units of time to enable the determination of this dynamics of contact with other users.

Several methods to determine ψ_i's distance from ψ_j using Bluetooth RSSI have been studied [37; 38; 39]. The proposed architecture does not predefine any fixed method to calculate *proximity* and leaves it at the discretion of the app providers to implement any method of their choosing. This method is independent of and can be changed without requiring a *fork* of the architecture.

E. Gateway adds data to a temporary block

The gateway, ξ_j, is essentially a *miner* with the added task of interfacing with users and anonymizing their data. When ξ_j receives an *upload-data* message from ψ_i at timestamp ψ_i^{TS}, ξ_j creates a *transaction* with data (ψ_i^{LAT}, ψ_i^{LAT}, ψ_i^{TS}, ψ_i^{PRN}, ψ_i^{F}), and adds the *transaction* to a temporary block in memory. The data in a *transaction* does not contain any user associated identity and can be added to the publicly-readable blockchain without compromising privacy.

It should be noted that although the terminology *transaction* is being used to be consistent with the blockchain literature, there is no scope or requirement of any actual transaction of any cryptocurrency.

F. Consortium attains consensus to add a block to the blockchain

The proposed architecture uses a *consortium* blockchain with the gateways knowing and partially trusting each other. The consortium needs to arrive at a consensus regarding which gateway should be permitted to add the next block to the blockchain.

Most common consensus algorithms' attributes make them unsuitable for use in the proposed architecture. For example, resources are wasted on computations unrelated to contact tracing in *Proof-of-Work*, cryptocurrencies are required to be wagered in *Proof-of-Stake*, and privacy is compromised by requiring users to upload data to multiple gateways simultaneously in *practical Byzantine Fault Tolerance* algorithm.

Hence, the proposed architecture uses a modified *Proof of Authority* consensus algorithm, where permission to add a block to the blockchain is granted to the gateways in a round-robin manner sorted on the IP addresses of the active gateways. This is a simple, efficient, and fair algorithm that uses *reputation* as a *stake*.

The reputation of a gateway, ξ_j is maintained by associating an *honesty-metric*, ξ_j^{HM}, with it. If any gateway, apart from the permitted one, attempts to add a block to the blockchain, then the rest of the gateways ignore the block and decrements the *honesty-metric* of the gateway by 1. The usage of *honesty-metric* to purge a malicious gateway from the consortium is further discussed in Section V-H.

TABLE III: Internal data structures at the gateways

Name	Structure	Location	Index	Data type	Data
HT	Hashtable	Main memory	ψ_i^{IP}	Tuple	$(\psi_i^{PUK}, \psi_i^{TS})$
DB_1	Key-value database	Persistent memory	ψ_i^{PUK}	Scalar	ψ_i^{IP}
DB_2	Key-value database	Persistent memory	ψ_i^{IP}	Stack of tuples	Each tuple in the stack contains $(\psi_i^{PRN}, \psi_i^{LAT}, \psi_i^{LNG}, \psi_i^{TS})$
DB_3	Key-value database	Persistent memory	ψ_i^{PRN}	Scalar	ψ_i^{PUK}
RT	Spatio-tempoaral database	Persistent memory	$(\psi_i^{LAT}, \psi_i^{LNG}, \psi_i^{TS})$	Tuple	$(\psi_i^{PRN}, \psi_i^{F})$

G. Gateway adds the temporary block to the blockchain

When the consensus algorithm chooses ξ_j to add a block to the blockchain, ξ_j adds the Merkel root hash of all the *transactions* in the temporary block to the block's header. Finally, the block is added to the blockchain with a hash-based link to the previous block.

The entire process, from a user initiating the registration process to the addition of a block to the blockchain, is shown in the sequence diagram in Fig. 3. Two users, ψ_0 and ψ_1, and three gateways, ξ_0, ξ_1, and ξ_2, are shown in the figure. ψ_0 was already registered and uploading data according to ψ_0's ordered set of the permuted active gateways (ξ_2, ξ_1, ξ_0) when ψ_1 started the registration process. ψ_1 obtains the list of known gateways, Φ_κ^1, from the public Internet. ψ_1 queries ξ_0, which is in Φ_κ^1, at timestamp t for Φ_α^t and ξ_0 responds accordingly. ψ_1 permutes Φ_α^t into the ordered set (ξ_1, ξ_2, ξ_0) and starts uploading data. Using a round-robin algorithm, both ψ_0 and ψ_1 uploads data to the gateways. Without loss of generality, ξ_1 is assumed to be chosen by the consensus algorithm, after which ξ_1 adds the next block to the blockchain.

H. Consortium validates the newly added block on the blockchain

After a block, B_n, is added to the blockchain by a gateway ξ_j, two other gateways, ξ_p and ξ_q, are chosen to validate the distribution of the *transactions* within the block. ξ_p is the gateway that appended the previous block, B_{n-1}, and is expected to have B_{n-1} in its main memory. ξ_q is chosen as follows.

To choose ξ_q as a validator, the gateways participate in a voting process constrained by two rules. (i) A gateway cannot vote for itself, (ii) A gateway can revote for any gateway only after exhaustively voting for all other active gateways. The two rules provide a fair majority voting scheme to choose ξ_q, with each gateway having no incentive to harbour bias. A gateway's adherence to these rules can be tracked by all other gateways in the consortium. Gateways that break the rules have their *honesty-metric* decremented by 1.

The gateways ξ_p and ξ_q divide the entire range of latitude and longitude data into discrete bins and creates two histograms based on the counts of *transactions* in blocks B_n and B_{n-1} corresponding to the bins. The *Chi-squared hypothesis test* is applied to validate that the distributions of the *transactions* in blocks B_n and B_{n-1} agree. The hypothesis test is performed twice - once on the entire histograms, and once only on the bins corresponding to the regions being served by ξ_j.

If both ξ_p and ξ_q accept the null hypothesis that *Both histograms in both tests come from the same distributions*, then ξ_j^{HM} is incremented by 1; else if both ξ_p and ξ_q reject the null hypothesis, then ξ_j^{HM} is decremented by 1.

Suppose there is a discrepancy in the Chi-squared test results of ξ_p and ξ_q, the consortium votes again to choose another gateway, ξ_r, for revalidation. The Chi-squared test result of ξ_r is accepted as truth, and ξ_j^{HM} is incremented or decremented by 1 based on ξ_r's outcome of the hypothesis test. Also, the gateway, ξ_p or ξ_q, whose Chi-squared test result contradicts with ξ_r's, have ξ_p^{HM} or ξ_q^{HM}, respectively, decremented by 1.

If ξ_j^{HM} of gateway ξ_j falls below 0, all other gateways in Φ broadcast *purge-gateway* messages to their registered users. If a user, ψ_i, receives *purge-gateway* message with ξ_j's IP address from more than half of the gateways in Φ_α^t, ψ_i removes ξ_j from Φ_α^t. Each gateway decrements $|\Phi_\alpha^t|$ in HT by 1, and the fraudulent gateway is removed from the consortium.

I. Gateway maintains data for contact tracing

A gateway, ξ_j, internally maintains the data structures given in Table III.

Timestamp ψ_i^{TS} corresponds to ψ_i's time of sending the most recent *upload-data* message. The timestamp must be consistent across all gateways and semantically follow the same timezone. This is facilitated by using the Unix Epoch time to set the value of the timestamp.

ξ_j updates the data structures HT, DB_1, DB_2, and DB_3, using Algorithm 1 whenever ξ_j receives any data packet from users in Ω. These data structures are internal to the gateways and are not shared by a gateway with any other actor in the network.

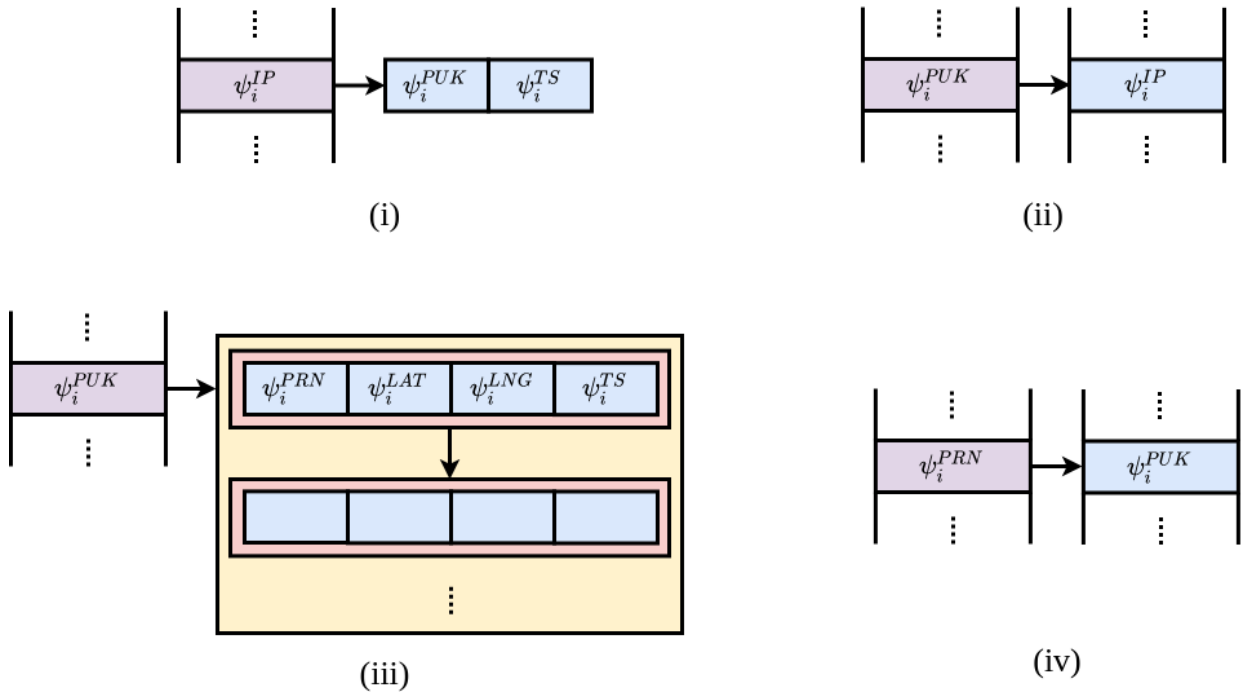

Fig. 4: Internal data at the gateways.
(i) HT, (ii) DB_1, (iii) DB_2, (iv) DB_3

The user of an IP address changes whenever a DHCP server assigns the IP address to a different user. A user, ψ_i, might be allocated the IP address $\psi_i^{IP} = \psi_j^{IP}$, which was previously assigned to another user ψ_j. When ξ_j receives a *request-to-join* message from ψ_i, it stores ψ_i^{IP} in DB_1, and reinitializes HT with ψ_i^{PUK} and current timestamp.

Algorithm 1: Maintaining internal data at the gateways

Input: Data Packet from IP address '`IP`'

Dependencies: Procedures `time` to retrieve current timestamp, and `send-active-gateway` to send Φ_α^t at time t

if $TYPE =$ *query-gateway* **then**
  `send-active-gateway()`;
else if $TYPE =$ *request-to-join* **then**
  $DB_1[\psi_i^{PUK}] \leftarrow$`IP`;
  HT[`IP`] $\leftarrow (\psi_i^{PUK}$, `time()`);
else if $TYPE =$ *upload-data* **then**
  $\psi_i^{PRN} \leftarrow HT$[`IP`].$\psi_i^{PRN}$;
  HT[`IP`].$\psi_i^{TS} \leftarrow$ `time()`;
  $DB_2[\psi_i^{PUK}]$.`push((`$\psi_i^{PRN}, \psi_i^{LAT}, \psi_i^{LNG}$, `time()))`;
  $DB_3[\psi_i^{PRN}] \leftarrow \psi_i^{PUK}$;

On the other hand, if ξ_j receives an *upload-data* message from ψ_i, ξ_j updates HT to set ψ_i^{TS} to the current timestamp, and updates DB_2 with payload data and DB_3 to associate ψ_i^{PRN} with ψ_i^{PUK}.

The RT Spatio-temporal database, built with a data structure like R-tree, stores public data from the blockchain for efficient retrieval of probable contacts in the vicinity of an at-risk or infected user. Whenever a block is added to the blockchain by any gateway, ξ_j stores $(\psi_i^{LAT}, \psi_i^{LNG}, \psi_i^{TS}) \rightarrow (\psi_i^{PRN}, \psi_i^{F})$ in RT for each of the *transactions* in the block.

J. Gateway carries out initial contact tracing

The initial contact tracing process, given in Algorithm 2, begins when a user $\rho \in P \cup C$ is diagnosed with the disease or comes in contact with a patient or a contact of a patient.

If ρ tests positive for the disease, then ρ sets $\rho^S =$ `0×F` before uploading all subsequent *upload-data* messages. Else if ρ is a contact, then ρ updates ρ^S and ρ^{CH} according to Algorithm 3. If the value of ρ^S changes, then ρ sets $\rho^{CH} =$ `1` for an entire iteration of the round-robin data uploading algorithm.

When a gateway, ξ_j, receives an *upload-data* message from ρ with ρ^{CH}=1, ξ_j initiates the contact tracing process. First, ξ_j looks up HT to obtain ρ^{PUK}, using which ξ_j queries DB_2 to retrieve ρ's travel history and set of generated PRNs. Then for each spatio-temporal location in ρ's travel history, ξ_j searches RT to obtain the set of all probable contacts who were in the vicinity of ρ. Finally, ξ_j obtains the IP address of each of the probable contact using DB_3 and DB_1, and alerts the probable contacts by sending *notify-contact* messages along with ρ^F and set of PRNs generated by ρ.

Algorithm 2: Initial contact tracing at the gateways

Input: *upload-data* or *keep-tracing* message

Dependencies: Procedures `findNeighbours` to search RT, and `send-notify-contact` and `send-keep-tracing` to send *notify-contact* and *keep-tracing* messages, respectively

if $TYPE = \textit{upload-data} \wedge \rho^{CH} = 1$ **then**
  $\rho^{PUK} \leftarrow HT[\rho^{IP}].\rho^{PUK}$;
  `travel_history` $\leftarrow DB_2[\rho^{PUK}]$;
  C^{PRN} = `[]`;
  for $(\rho^{LAT}, \rho^{LNG}, \rho^{TS}) \in$ `travel_history` **do**
    C^{PRN}.append(findNeighbours($(\rho^{LAT}, \rho^{LNG}, \rho^{TS})$));
  end
  UC^{PRN} = `[]`;
  for $c^{PRN} \in C^{PRN}$ **do**
    $c^{PUK} \leftarrow DB_3[c^{PRN}]$;
    $c^{IP} \leftarrow DB_1[c^{PUK}]$;
    if `search-miss` **then**
      UC^{PRN}.append(c^{PRN});
    end
    send-notify-contact($c^{IP}, \rho^{F}, \rho^{PRN} \in$ travel_history);
  end
  send-keep-tracing($\rho^{F}, \rho^{PRN} \in$ travel_history, UC^{PRN});
else if $TYPE = \textit{keep-tracing}$ **then**
  for $c^{PRN} \in$ *message payload* **do**
    $c^{PUK} \leftarrow DB_3[c^{PRN}]$;
    $c^{IP} \leftarrow DB_1[c^{PUK}]$;
    send-notify-contact($c^{IP}, \rho^{F}, \rho^{PRN} \in$ payload);
  end

Since ξ_j receives infrequent data from the users, it would be able to retrieve only a partial travel history of ρ and a partial set of the IP addresses of the probable contacts. ξ_j broadcasts *keep-tracing* messages to the other active gateways with ρ's FLAG (ρ^F), the set of PRNs generated by ρ, and the set of PRNs of the probable contacts whose IP addresses are unknown to ξ_j. Without compromising privacy, this anonymised data packet enables other gateways to alert the contacts not known to ξ_j.

When a gateway, ξ_k, receives a *keep-tracing* message from ξ_j, it retrieves the IP addresses of all the known potential contacts encoded in the message, and alerts the known potential contacts using *notify-contact* messages.

Because of privacy preservation measures, each gateway in the consortium can only be partially involved in tracing a user's contacts. However, all gateways in the consortium collectively enable complete contact tracing.

K. User eliminates false positives in contact tracing

When a potential contact, c_i, receives the set ρ^{PRN} from ξ_j, c_i retrieves the *proximity* associated with each PRN in ρ^{PRN} from BT. The c_i^S and c_i^{CH} in c_i's subsequent *upload-data* messages are updated using Algorithm 3. If the value of c_i^S changes, c_i^{CH} is set to 1 for a complete iteration of the round-robin data upload algorithm.

If a gateway, ξ_j, detects c_i^{CH} to be set to 1, ξ_j starts the contact tracing process again to find the contacts of c_i. If c_i was a primary contact of an infected patient, ρ, the second iteration of the contact tracing process will find the secondary contacts of ρ. This algorithm iteratively finds all possible contacts of ρ, forming the complete set of probable contacts C.

L. GIS data analysis

The data in the *transactions* in a block of the blockchain are in unsorted linear order, and the ones in RT are in a multidimensional-indexed trees, essentially *sorted* using the index (ψ_i^{LAT}, ψ_i^{LNG}, ψ_i^{TS}). Although the users' generated PRN in these data structures are pseudo-random identifiers and have no semantic meaning, the other fields, (ψ_i^{LAT}, ψ_i^{LNG}, ψ_i^{TS}, and ψ_i^{S}) can be analysed to reveal trends and patterns, and make predictions on the spread of the disease.

A few examples of such analysis on the data in RT are identification of disease clusters, observation of the rate of increase or decrease of cases locally and globally, and tracking

the path of the spread of the disease. A data analyst can use the anonymised data on the publicly-readable blockchain for innumerable other use cases.

The following subsections illustrate two of the possible GIS analysis on the data in the Spatio-temporal tree.

Algorithm 3: Update c_i^S and c_i^{CH} of a potential contact c_i

Input: c_i^S: Current status bits $(S_3 - S_0)$ of c_i

Input: ρ^S: Status bits of $(S_3 - S_0)$ ρ received from ξ_j

Input: L_{BT}: List of *proximity* retrieved from BT corresponding to ρ^{PRN}

Output: c_i^S and c_i^{CH} for c_i's subsequent data packet

Dependencies: Procedures `bitsToHex` and `hexToBits` to convert the four-bit status flags to an unsigned half byte and vice-versa, respectively.

$c_i^{S_{hex}}$, $\rho^{S_{hex}} \leftarrow$ `bitsToHex`(c_i^S), `bitsToHex`(ρ^S);
$risk \leftarrow \max_{a \in L_{BT}} (\frac{a \times \rho^{S_{hex}}}{0 \times F})$;
$c_i^{S_{new-hex}} \leftarrow risk + c_i^{S_{hex}}$;
if *sum overflow* $\vee$ $c_i^{S_{new-hex}} = 0{\times}F$ **then**
  $c_i^{S_{new}} \leftarrow 0{\times}$E;
else
  $c_i^{S_{new}} \leftarrow$ `hexToBits`($c_i^{S_{new-hex}}$);
end
$c_i^{CB_{new}} \leftarrow 1$;
if $risk = 0{\times}0$ **then**
  $c_i^{CB_{new}} \leftarrow 0$;
end
c_i^S, $c_i^{CH} \leftarrow c_i^{S_{new}}$, $c_i^{CH_{new}}$

1) Identification of disease clusters: To identify disease clusters at a particular time in a region determined by a bounding box, bin the entire region into smaller sub-regions. For example, a 30×30 sq km regions can be segmented into 900 sub-regions of 1×1 sq km each. Each of the sub-region is defined by a bounding box. The multi-dimensional Spatio-temporal tree is queried with each sub-region geographic bounding box and a time interval of $\pm\frac{\tau}{2}$ units of time. The time interval ensures that exactly one uploaded data for each user is retrieved.

The `STATUS BITS` of ψ_i^F for each data tuple retrieved from the tree for each sub-region is parsed to count the infected patients. The total count of infected users of the sub-region, along with the latitude and longitude corresponding to the centroid of the sub-region, is stored in a dataset for further analysis.

In the absence of a standard definition of a disease hotspot, analysts of different regions are free to choose how hotspots are defined. The methodology used by Uttar Pradesh and Delhi government, wherein they use a threshold of 6 cases in an area to define a hotspot, is used. All data points with a count less than the threshold are eliminated from the dataset. Thus, only the centroids corresponding to disease hotspots remains on the dataset.

Finally, a clustering algorithm is applied to find disease clusters on the dataset. Since the number of probable clusters is unknown beforehand, algorithms like K-Means cannot be used. Hence, DBSCAN is applied to the dataset to find the various clusters in the region.

2) rate of change of local infection cases: The rate of change in the number of cases can be evaluated and visualised for any region, determined by a bounding box. The duration and granularity of the analysis determine how the analysis is to be undertaken.

Suppose that an analyst wishes to evaluate the rate of change of cases for a duration of 30 days with a granularity of 3 days. Then the Rtree is queried $\frac{30}{3} = 10$ times with the spatial indices of the bounding box of the region and the temporal index as determined by the 10 time quanta.

The data retrieved in each of the 10 queries are parsed to determine the number of data tuples associated with infected users. However, a single infected user would have uploaded data multiple times during the time quantum of evaluation. He or she would be counted multiple times while determining the count of infected users. Therefore, the count is normalized by dividing it by $\frac{3 \text{ days}}{\tau \text{ units of time}}$, where 3 days is the time quantum determined for the analysis.

Because of the anonymisation of data, tracking and counting each user precisely is not possible. Thus, 10 real-valued numbers indicating are obtained using the aforementioned methodology to approximate the count of infection cases in a region for a pre-determined time period and granularity. This can be plotted on a geographic map or as a graph to visually show the rate of change of count of cases.

VI. Implementation

In lieu of a large number of actual users, a Python script was used to simulate users generating synthetic data. Three different simulations were undertaken on various scales: small town, large city, and state (province). Table IV shows the summary of the simulations.

The data generated was confined to the geographic location of Assam, India and Abuja, Nigeria, for ease of development. Certain number of *home*, *office* and other hotspots (restaurants, markets, etc) were randomly pre-determined. A simulated user is randomly assigned a home and an office during initialisation. A maximum of 8 users is assigned a single home address. The user starts the day at home, goes to the office, and returns home in the evening. At different times of the day, the user may randomly visit other hotspots (with a probability 0.1). Figure

TABLE IV: Types of simulations

Simulation Type	Population	Area (sq. kms.)	Count of homes	Count of offices	Count of hotspots
Small town	5×10^4	10^2	10^4	10^2	10^2
Large City	9.5×10^5	1.6×10^4	2.47×10^5	10^3	5×10^2
State	10	20	30	40	50

X shows the path taken by a user for 3 days, and Figure Y shows the paths for 10 users on a single day.

A prototype system with gateways is implemented using Python and deployed in Docker containers in a `HP Z820` workstation. Multiple Docker instances in a Docker swarm were used to simulate a consortium of up to 256 gateways. The `Rtree` Python module [40] was used to implement the multi-indexed Spatio-temporal tree. The gateways' implementation of the blockchain is based on [41], with the consensus algorithm known as Proof-of-Elapsed-Time[42].

VII. Discussion

The following characteristics of the proposed architecture address the criticisms enumerated in Section I.

1) **Interoperability:** The blockchain technology enables a globally consistent database that can be used to create interoperable contact tracing apps.
2) **Resiliance to DoS attacks:** usage of multiple gateways reduces the probability of a DoS attack making the database unavailable.
3) **Data immutability:** Database is append-only.
4) **Privacy preservation:** Data anonymisation dissociates the user's identity from the uploaded data. The identity of the user is distributed across all gateways.
5) **Transparency:** The blockchain can be read by the public. It can be analysed to visualise global and local trends and patterns, and by users for self contact tracing without compromising user identity.
6) **Trustlessness:** The user need not trust any centralised authority.
7) **Multimodal data acquisition:** Data is gathered through a smartphone's GPS sensors and Bluetooth tokens. These are used in conjunction for a two-phase contact tracing process, eliminating false negatives.

The degree of privacy provided to users depends on the globally consistent value of τ. A too low value τ would increase traffic in the network, increase load at the gateways and reduce privacy. Conversely, a too high value of τ would reduce the data granularity, restricting the accuracy of the contact tracing process.

The value of τ adjusts dynamically to restrict the exposure of users' data to a gateway to no more than once per 12 hours. It can be calculated that if $|\Phi_\alpha^t| = 144$, then $\tau = 5$ minutes; and if $|\Phi_\alpha^t| = 720$, then $\tau = 1$ minute.

The architecture addresses the three identified notions of privacy in contact tracing.

1) ***Privacy from snoopers*** The user signs and encrypts all location and infection status data before uploading it to gateways.
2) ***Privacy from contacts*** The contacts use pseudo-random identifiers to analyse their risk of exposure. The identity of a user is hidden from other users.
3) ***Privacy from the authorities*** Each gateway can identify ψ_i's data once every $\tau \times |\Phi_\alpha^t|$ units of time. The latency increases with the increase in the number of gateways. Active tracking of ψ_i's identity is strenuous with such low sampling frequency.

A gateway, ξ_j, expects to receive data from a user, ψ_i, once every 12 hours. If ψ_i uploads data at a higher frequency, ξ_j blocks ψ_i's IP address by a firewall rule until τ units of time before ψ_i's next valid allotted timestamp. Hence, this prevents a user from flooding the network.

The architecture expects the gateways to be deployed on parallel, high-performance infrastructure with load balancers. If the architecture is used by a significant portion of the $3{\times}10^9$ smartphone users globally, each gateway would be required to process around $69{\times}10^3$ *upload-data* messages per second on average.

The internal data structures maintained at the gateways can be pruned to save storage costs. If the contact tracing uses historical data of, say, 14 days, all data older than 14 days can be removed from the data structures. This would constrain the stack corresponding to each user's data in DB_2 to around 1.2kB in size.

One unaddressed issue in the proposed architecture is the *incentive* for the gateways. The proposed architecture assumes that countries and organisations will authorise and host gateways for the sake of public health. The introduction of a cryptocurrency into the architecture could provide a monetary incentive for gateways, and remove the dependency on the benevolence of the aforementioned countries and organisations.

The cryptocurrency could be mined when gateways add blocks to the blockchain, and used when non-gateways attempt to read the blockchain for contact tracing or analysis. Also, the existence of a cryptocurrency would allow the usage of the *proof-of-stake* consensus algorithm to deploy a public blockchain.

Another issue in the architecture is the gateways' trust in the users' uploaded data and risk assessment. The use of a proof-publishing oracle can remove the trust. However, oracles have not been used in the proposed architecture because it increases authorities' role in the network, thus partially compromising privacy.

VIII. Conclusion

The COVID-19 pandemic has necessitated extensive use of digital contact tracing apps. Predictively, multiple

non-interoperable apps have been developed and deployed in many countries. However, many of these apps have attracted criticism for privacy and security issues. This paper proposes an architecture that provides a publicly-readable and globally consistent blockchain-based database for contact tracing app providers. The publicly-readable blockchain can be used for transparent contact tracing and GIS data analysis to reveal global and local trends and patterns.

The reliability of data is enabled by the immutability of the blockchain. Data anonymisation disassociates the user's identity from their data on the blockchain. The three notions of privacy - (1) privacy from snoopers, (2) privacy from contacts, and (3) privacy from the authorities, are addressed by the proposed architecture. The architecture distributes the users' identities among the gateways, thereby restricting an authority's ability to track a user. Moreover, because of the usage of encryption and pseudo-random identifiers, other actors in the system cannot intercept and snoop a user's location and infection status data.

Apart from privacy preservation and security, interoperability is enabled among the multiple apps using the architecture because of the usage of consistent message formats and a shared blockchain.

DISCLOSURE STATEMENT

We wish to confirm no known conflicts of interest associated with this publication. There has been no significant financial support for this work that could have influenced its outcome. Furthermore, no financial interest or benefit has arisen from the direct applications of the research.

FUNDING

No funding was received for this work.

REFERENCES

[1] World Health Organization. WHO Coronavirus Disease (COVID-19) Dashboard. World Health Organization; 2020. Available from: https://covid19.who.int/.

[2] Klinkenberg D, Fraser C, Heesterbeek H. The Effectiveness of Contact Tracing in Emerging Epidemics. PLoS ONE. 2006 Dec;1(1):e12. Available from: https://dx.plos.org/10.1371/journal.pone.0000012.

[3] Salath M, Althaus CL, Neher R, Stringhini S, Hodcroft E, Fellay J, et al. COVID-19 epidemic in Switzerland: on the importance of testing, contact tracing and isolation. Swiss Medical Weekly. 2020 Mar. Available from: https://doi.emh.ch/smw.2020.20225.

[4] Ferretti L, Wymant C, Kendall M, Zhao L, Nurtay A, Abeler-Dörner L, et al. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. Science. 2020 May;368(6491):eabb6936. Available from: https://www.sciencemag.org/lookup/doi/10.1126/science.abb6936.

[5] Raper J, Gartner G, Karimi H, Rizos C. A critical evaluation of location based services and their potential. Journal of Location Based Services. 2007 Mar;1(1):5-45. Available from: http://www.tandfonline.com/doi/abs/10.1080/17489720701584069.

[6] Bridwell SA, Miller HJ. Location-Based Services. In: Liu L, Özsu MT, editors. Encyclopedia of Database Systems. Boston, MA: Springer US; 2009. p. 1639-41. Available from: http://link.springer.com/10.1007/978-0-387-39940-9_494.

[7] Bahri S. Enhancing quality of data through automated SARS contact tracing method using RFID technology. International Journal of Networking and Virtual Organisations. 2007;4(2):145. Available from: http://www.inderscience.com/link.php?id=13540.

[8] Farrahi K, Emonet R, Cebrian M. Epidemic Contact Tracing via Communication Traces. PLoS ONE. 2014 May;9(5):e95133. Available from: https://dx.plos.org/10.1371/journal.pone.0095133.

[9] Jhunjhunwala A. Role of Telecom Network to Manage COVID-19 in India: Aarogya Setu. Transactions of the Indian National Academy of Engineering. 2020 Jun. Available from: http://link.springer.com/10.1007/s41403-020-00109-7.

[10] Stevens H, Haines MB. TraceTogether: Pandemic Response, Democracy, and Technology. East Asian Science, Technology and Society. 2020.

[11] Abubakar A, Babate AI, Ishola A, Sani AM. The Study of Data Security in Cloud Computing. European Journal of Electrical Engineering and Computer Science. 2020;4(4).

[12] Cho H, Ippolito D, Yu YW. Contact tracing mobile apps for COVID-19: Privacy considerations and related trade-offs. arXiv preprint arXiv:200311511. 2020.

[13] Kim M, Denyer S. A'travel log'of the times in South Korea: Mapping the movements of coronavirus carriers. The Washington Post. 2020.

[14] Troncoso C, Payer M, Hubaux JP, Salathé M, Larus J, Bugnion E, et al. Decentralized privacy-preserving proximity tracing. arXiv preprint arXiv:200512273. 2020.

[15] Yasaka TM, Lehrich BM, Sahyouni R. Peer-to-Peer Contact Tracing: Development of a Privacy-Preserving Smartphone App. JMIR mHealth and uHealth. 2020 Apr;8(4):e18936. Available from: https://mhealth.jmir.org/2020/4/e18936.

[16] Altuwaiyan T, Hadian M, Liang X. EPIC: efficient privacy-preserving contact tracing for infection detection. In: 2018 IEEE International Conference on Communications (ICC). IEEE; 2018. p. 1-6.

[17] Brack S, Reichert L, Scheuermann B. Decentralized Contact Tracing Using a DHT and Blind Signatures. IACR Cryptol ePrint Arch. 2020;2020:398.

[18] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. Manubot; 2019.

[19] Buterin V, et al. Ethereum white paper: a next generation smart contract & decentralized application platform. First version. 2014;53.

[20] Cachin C, et al. Architecture of the hyperledger blockchain fabric. In: Workshop on distributed

cryptocurrencies and consensus ledgers. vol. 310-4; 2016. .

[21] Kumar N, Aggarwal S, Sharma PR. Blockchain technology for secure and smart applications across industry verticals. S.l.: Elsevier Academic Press; 2021. OCLC: 1164493410.

[22] Chaudhary R, Jindal A, Aujla GS, Aggarwal S, Kumar N, Choo KKR. BEST: Blockchain-based secure energy trading in SDN-enabled intelligent transportation system. Computers & Security. 2019 Aug;85:288-99. Available from: https://linkinghub.elsevier.com/retrieve/pii/S016740481831201X.

[23] Mistry I, Tanwar S, Tyagi S, Kumar N. Blockchain for 5G-enabled IoT for industrial automation: A systematic review, solutions, and challenges. Mechanical Systems and Signal Processing. 2020 Jan;135:106382. Available from: http://www.sciencedirect.com/science/article/pii/S088832701930603X.

[24] Aggarwal S, Chaudhary R, Aujla GS, Kumar N, Choo KKR, Zomaya AY. Blockchain for smart communities: Applications, challenges and opportunities. Journal of Network and Computer Applications. 2019 Oct;144:13-48. Available from: http://www.sciencedirect.com/science/article/pii/S1084804519302231.

[25] Sharma PK, Kumar N, Park JH. Blockchain-Based Distributed Framework for Automotive Industry in a Smart City. IEEE Transactions on Industrial Informatics. 2019 Jul;15(7):4197-205.

[26] Jindal A, Aujla GS, Kumar N. SURVIVOR: A blockchain based edge-as-a-service framework for secure energy trading in SDN-enabled vehicle-to-grid environment. Computer Networks. 2019 Apr;153:36-48. Available from: http://www.sciencedirect.com/science/article/pii/S138912861831106X.

[27] Farnaghi M, Mansourian A. Blockchain, an enabling technology for transparent and accountable decentralized public participatory GIS. Cities. 2020;105:102850.

[28] Tian F. An agri-food supply chain traceability system for China based on RFID & blockchain technology. In: 2016 13th international conference on service systems and service management (ICSSSM). IEEE; 2016. p. 1-6.

[29] Keirns G. Blockchain Land Registry Tech Gets Test in Brazil. CoinDesk News. 2017;5:2017.

[30] Chavez-Dreyfuss G. Honduras to build land title registry using bitcoin technology. Reuters Disponível em:¡ http://in reuters com/article/usa-honduras-technology-idINKBN0O01V720150515. 2015.

[31] Oprunenco A, Akmeemana C. Using blockchain to make land registry more reliable in India. LSE Business Review. 2018.

[32] Kim C. Sweden's land registry demos live transaction on a blockchain; 2019.

[33] Torun A. Hierarchical blockchain architecture for a relaxed hegemony on cadastre data management and update: A case study for Turkey. In: Proceedings of the UCTEA International Geographical Information Systems Congress; 2017. .

[34] Bansal A, Garg C, Padappayil RP. Optimizing the Implementation of COVID-19 "Immunity Certificates" Using Blockchain. Journal of Medical Systems. 2020;44(9):1-2.

[35] Lv W, Wu S, Jiang C, Cui Y, Qiu X, Zhang Y. Decentralized Blockchain for Privacy-Preserving Large-Scale Contact Tracing. arXiv preprint arXiv:200700894. 2020.

[36] Xu H, Zhang L, Onireti O, Fang Y, Buchanan WB, Imran MA. BeepTrace: Blockchain-enabled Privacy-preserving Contact Tracing for COVID-19 Pandemic and Beyond. ArXiv. 2020;abs/2005.10103.

[37] Jung J, Kang Do, Bae C. Peer to peer signal strength characteristic between IoT devices for distance estimation. In: 2014 IEEE World Forum on Internet of Things (WF-IoT). IEEE; 2014. p. 208-11.

[38] Huang J, Chai S, Yang N, Liu L. A novel distance estimation algorithm for Bluetooth devices using RSSI. In: 2017 2nd International Conference on Control, Automation and Artificial Intelligence (CAAI 2017). Atlantis Press; 2017. .

[39] Zhou S, Pollard JK. Position measurement using Bluetooth. IEEE Transactions on Consumer Electronics. 2006;52(2):555-8.

[40] Gilles S, Butler H. Spatial indexing for Python; 2019. Available from: https://rtree.readthedocs.io/en/latest/.

[41] Kansal S. Develop a blockchain application from scratch in Python; 2020. Available from: https://developer.ibm.com/technologies/blockchain/tutorials/develop-a-blockchain-application-from-scratch-in-python/.

[42] Pal A, Kant K. DC-PoET: Proof-of-Elapsed-Time Consensus with Distributed Coordination for Blockchain Networks. In: 2021 IFIP Networking Conference (IFIP Networking). IEEE; 2021. p. 1-9.