Contact Tracing Solution for Global Community

HARI T.S. NARAYANAN

NetTools Consulting, 6/13, Gokulam Colony, Chennai 600033, India

ts.hari@gmail.com

There are several Contact Tracing solutions since the outbreak of SARS COVID-19. All these solutions are localized—specific to a country. The Apps supported by these solutions do not interwork with each other. There are no standards to the proximity data collected by these Apps. Once the international travel restrictions are relaxed, this will become an issue. This paper explores this issue, by addressing one of the key requirements of Contact Tracing solutions. All the current solutions use an Identifier, Proximity Identifier (PID), that anonymously represents the user in the proximity data exchanged. The PID used in these applications varies in their structure, management and properties. This paper first identifies the common desirable properties of PID, including some non-obvious ones for its global application. This identification is essential for the design and development of the Contact Tracing solution that can work across boundaries seamlessly. The paper also evaluates representative solutions from two different design classes against these properties.

Keywords: contact tracing; interworking; SARS COVID-19; Trace Together; exposure notification; Mobile Apps; proximity identifier; global solution; Bluetooth low energy

Received 17 July 2020; Revised 1 February 2021; Editorial Decision 28 May 2021
Handling editor: Fabrizio Messina

1. INTRODUCTION

A Contact Tracing system collects, stores and processes data to identify the contacts and the cases for an infection. A subject is a case when the subject is having infection-specific symptoms and signs. A subject is a contact if the subject is likely infected but not having any symptoms or signs yet. A contact or a case happens when a subject is in the proximity of an infected person. This event could be physical contact, close contact or proximity contact for Contact Tracing purpose. The physical contact includes exchange of bodily fluids. The close contact or simply contact is described by the guidelines released and periodically updated by the Centers for Disease Control (CDC) and Prevention, US [1]. Almost all current Contact Tracing initiatives [2–7] are built with Bluetooth Low Energy (BLE) technology [8] present in smartphones. These solutions are designed differently, and their operations are localized to respective countries. At present, with international travel restrictions in place, the localized Contact Tracing solutions [2–7] are adequate. Once these restrictions are lifted then these non-interworking solutions may find to be inadequate.

A person traveling to country A from country B needs to install the App used in A on landing. If this subject gets infected in country A, then the proximity data collected in the phone needs to be shared and processed by Contact Tracing initiatives of both countries. The data collected by the two Apps need to be processed and consumed in their respective countries only. This means that the data in the subject’s phone need to be segregated and sent over a network to appropriate healthcare authorities in country B securely. There are some social and logistical issues in this. The social issues are privacy related; the privacy law varies across countries. Even though it is possible to accomplish the transfer of data, the delay in sorting out the issues with manual intervention could cost in terms of precious time. The problem gets more complicated if the person visits multiple countries and is found to be infected when returning to the country of origin.

One way to address the solution is by creating a well-defined process for the exchange of proximity data among all the countries, then use this process to exchange data. This is a complex time-consuming initiative considering the non-linear nature of the activity. The other option is technical and creates a Contact Tracing solution that works seamlessly with international travel activity. This paper dwells on the latter of the two. A complete solution needs initiatives on both aspects. The content of the proximity message and its representation needs to be standardized. The data exchanged in current proximity messages are
not the same. One problem that is addressed in this paper is related to the Proximity Identifier (PID). The PID anonymously represents the subject in proximity records. The properties and management of this PID are quite different across current contact tracing applications. This paper studies the desirable properties of this identifier and their implications to Contact Tracing design. Some of these properties are obvious and some are identified with the issues found with the current designs. This analysis should eventually help in defining the right kind of PID that meets the requirements of a seamlessly working global Contact Tracing solution like the one presented here [9].

The rest of the paper is divided into seven Sections. Section 2 enumerates some obvious properties of PID. Section 3 describes two distinct types of Contact Tracing solutions that are currently in use, centralized and distributed. Sections 4 and 5 make the core part of this paper. These two Sections review the operations built around the PIDs to see how these properties manifest and how they influence the performance and resource requirements of the two solution types. These two sections also throw more light on some new desirable properties of PID and areas of issues. Section 6 completes the set of these properties with newly observed ones. Section 7 concludes the paper.

2. PROPERTIES OF PID

This paper starts with a list of obvious PID properties and updates this list after completing the review of the currently deployed Contact Tracing solutions. This is an iterative process, open for additional properties as our understanding of the issues get better. A pragmatic approach is taken here rather than a formal approach to prove completeness and independence of these properties.

1. **Unambiguous**: A given PID should unambiguously map to a single subject. A subject can have multiple PIDs, each one of these PIDs should map to a single subject. There can be multiple PIDs for the same subject and multiple types of PID in a solution to play different roles.

2. **Large Domain Space**: PID domain space should be large enough. There should be enough PIDs required to cover all the subjects to be contact traced.

3. **Anonymous**: A given PID should be anonymous. The PID should not reveal any of the characteristics (nationality, gender, etc.) or Identifiers (mobile ID, email ID, etc.) of the subject during contact tracing operation. Anonymity requirement is the reason proximity distance is used instead of location.

4. **Non-trackable**: A given PID should not be wireless trackable for more than a certain period.

5. **Decodable**: A given PID should be easily decodable when needed—given the PID, the authenticated and authorized person should be able to find and locate the associated subject without unreasonable administrative or procedural delay.

6. **Robust**: The PID and the associated procedures should be robust enough to handle security attacks and network outages.

7. **Constraint-Free**: The procedures associated with the PID should not have constraints that suspend or hinder Contact Tracing operation.

8. **Affordable Resource Requirement**: The resources (bandwidth, processing and storage) required for all PID-related operations should be affordable.

The first three properties are more crisply defined than the latter ones. The latter properties have certain fuzziness associated with their description. For instance, the non-trackable property suggests that the subject is non-trackable beyond a certain period. The value of this period is decided at design time as per some guidelines. This means that these properties are required to a certain level of desired degree. On the other hand, the first three are mandatory and critical.

3. CONTACT TRACING APPLICATIONS

All currently deployed Contact Tracing solutions are built with smartphones. Some solutions offer wearable options [4]. The BLE present in smartphones is used in exchanging proximity data. There are solutions where this is complemented by tracking the subject’s location using cellular network or Global Positioning System or both. The exchanged data are processed when needed or periodically to identify the contacts. Once a contact is found, the contact is recommended either quarantine or isolation. Every significant Contact Tracing operation is built around its PID.

There are two approaches to exchange proximity data with BLE; it is done either using intrusive or non-intrusive methods. In the intrusive approach, two smartphones set up a BLE connection and get the proximity data exchanged. In the non-intrusive approach, the data are scanned and extracted from the periodic advertisement messages sent out by BLE adapter in smartphones. The intrusive design is limited by its scalability and reliability in scenarios where there are many smartphones within Bluetooth adapter’s range. Transit points, protests, celebrations, religious and sports gatherings are examples of such scenarios. The Contact Tracing solution is expected to be reliable in these conditions for it to be effective. The intrusive design also requires more resources, specifically power. Non-intrusive approach is light weight compared with the intrusive one, but the amount of data exchanged is limited to 31-bytes at present. Irrespective of the method, the messages exchanged periodically are used to compute proximity duration and proximity distance.

There are two approaches to processing proximity data. In one approach, the data are uploaded and processed in a centralized server. In the second approach, the data are processed in every registered smartphone, thus distributing the processing load. The latter approach may also make use of...
Contact Tracing Solution for Global Community

4. THE FOUR OPERATIONAL PHASES OF TRACETOGETHER

The TraceTogether uses an authenticated, encrypted and rotating Proximity ID (TempID). User needs to complete a registration procedure with a server at App installation time. This server is administered by the Singapore Ministry of Health.

4.1. App/user registration phase

The registration operation for TraceTogether starts after the user installs the App on the user’s smartphone. User completes the installation and starts the registration process with the server that is hardwired into the App. The server validates the App and the user, and then adds the phone number to a database that maintains the user’s participation in Contact Tracing. The database may also include many other user-related data besides the phone number. These other data items are outside the scope of this paper. The server generates a random, unique 21-byte ID, UserID, and maps the ID to the user’s phone number. This mapping is used later in locating the user when needed. This UserID is the PID for TraceTogether initiative. In proximity data exchange, a rolling TempID with encryption and message digest is used. This 84-byte TempID includes UserID, Lifetime of TempID and other security parameters as shown in Fig. 1. The reverse mapping, PID to user phone number, can be performed at the data centre with a different functional entity. The UserID and Lifetime fields are encrypted with a single common key that is maintained at the server. The Authentication (Auth) Tag supports message integrity, and the Initialization Vector (IV) supports robust data stream security. The UserID is used for mapping; and the TempID built with it is used for proximity data transport and exchange. This means that there are two types of PID: one to create an anonymous identifier by mapping and one for exchange operation. The 84-byte TempID is too large to employ non-intrusive methods for the exchange of proximity data.

4.2. PID management phase

The management of TempID may be handled by a functional entity (Server) that is different from the one that handles the registration. The TempID (RPID) is encrypted and decrypted with the common key administered at the Server. The TempID is rotated every 15 minutes (Fig. 2).

Every registered App gets its new TempID periodically. The lifetime of TempID is specified by the start and expiry time. Both are encoded within the TempID. The TempID cannot be exchanged before its start time and cannot be used after its expiry time. The start and expiry times are specified using Unix Epoch Time [10]. A batch of TempIDs could be sent in a single transfer to handle connectivity outages.

4.3. Proximity data exchange and logging phase

Once the registration is done, the App starts exchanging the proximity data with the peer Apps, the set of same Apps running
in other smartphones. The proximity data exchanged includes the TempID of the transmitting phone and the signal strength at the source (or transmitting device detail). At the receiving side, timestamp and the signal strength (Received Signal Strength Indicator—RSSI) are added to proximity data before logging the data.

4.4 Proximity data upload and processing phase

If one of the subjects is infected, the healthcare authorities with the consent of the subject upload the Proximity Data (TempID, RSSI, Timestamp and the Signal Strength at the source) from the subject’s smartphone to the server. These uploaded data include the UserIDs of all those subjects who had contact with the case in the last 25 days. The data for the last 14-days (incubation period of SARS-2) are sufficient for identifying contacts. However, TraceTogether logs the data for the additional 11 days to support other requirements. The term contact is defined differently in TraceTogether: close contact (less than about 6 feet, for at least 30 minutes), casual contact (less than about 6 feet, for at least 5 minutes but less than 30 minutes) and transient contact (less than about 6 feet, but less than 5 minutes). The UserIDs of the contacts need to be mapped to their respective mobile numbers for recommending quarantine or isolation. This mapping is achieved with the mapping table created at the time of registration.

4.5 Evaluating the PID of TraceTogether

The operational description from the earlier part of this Section provides a context to evaluate the PID of TraceTogether:

1. Unambiguous: The UserID (21-bytes) is chosen from a large pool of numbers. The size of this pool is $\approx 10^n$. Once a number is chosen, it is likely taken out of the pool to make sure that the UserID stays mapped to a single subject in the Server’s mapping database. This mapping is used later in identifying potential contacts and cases. TempID (84-bytes) is used in proximity data exchange. The TempID includes encrypted UserID, validation period and security parameters.

2. Domain Size: The size of the UserID pool is $\approx 10^n$. This is more than enough to cover the world population of close to 8 billion even with multiple assignments.

3. Anonymous: There is nothing in the TempID that is indicative of the subject’s characteristics: Country, ethnicity, location, age, gender, etc. Identifiers: Name, Mobile Number, IPv6 Address, Email ID. However, the UserID in the encrypted section of the TempID is free to have some value encoded to it. The use of dual Identifiers offers this freedom.

4. Non-Trackable: The TempID is rotated every 15-minutes making it difficult to Wirelessly track the subject beyond 15 minutes. There is still a possibility for tracking over a longer duration when a batch of TempIDs of a single individual is captured with the Man in the Middle attack.

5. Decodable: Once a UserID is known, the corresponding phone could be identified using the mapping created during the registration operation. The TempID is authenticated and the UserID is extracted from it with a single common key (common to all the TempIDs of all the users) administered at the server. This could be done only at the data centre, by the authorized application. One thing that needs validation is the operational issues to complete this reverse mapping. This validation is presented in the later part of this Section.

6. Robust: This UserID is authenticated, encrypted and rotated every 15 minutes. A single key is used to encrypt UserID of all the users to avoid key management problems. Once this common key is compromised, then the entire system is exposed. This could disrupt the operation of the system with a possible data loss. Transition to a new key is not documented in the design. A hacker can capture a TempID and impersonate with that to create false positive incidences for a short period of less than 15 minutes.

7. Constraint-and-Limitations: If there is no Internet connection, the periodic delivery of TempID could be interrupted and the App needs to suspend transmission of its proximity data. Batched TempIDs are used to address this problem. The BLE advertisement message is not big enough to support the 84-byte TempID and thus limits the design to intrusive data exchange. The Bluetooth 5.0 supports larger (244-bytes) payloads and addresses this limitation. However, Bluetooth 5.0 is not a viable option for Contact Tracing yet. The BLE handshake is constrained by the number of exchanges in certain scenarios.

4.6 Affordable resource requirement—validation

An operational analysis is needed to show that all major operations in this design are accomplished using affordable resources. An informal analysis is used in completing this sub-section. Unlike other properties, the evaluation of this property is unevenly large. A resource model is needed to show that the required resources (Computing Cycles, Network Bandwidth and Storage) are affordable. The following data are used to compute the resource requirements for various operations with CDC guidelines:

- The number of new cases for the USA was close to 75,000 for 16 July 2020; these data are from US COVID-19 statistics [11]. Assumption: these cases are spread over the 24-hour period uniformly. That is close to 6500 new cases for every 2 hours on the average. This
assumption eases the computational requirements. The actual number of cases can have large variance around this average value.

• The number of smartphones in the USA is close to 275 million [12]—which is assumed to be the number of instances of contact tracing Apps.

• Number of proximity consolidated records (PR) logged in a smartphone per day is about 150. This number could vary based on various factors—location, population density, etc. Identifying the range and the associated characteristics of this value require further research. The consolidated records are created with raw records exchanged. The number of raw records could be much larger than consolidated records depending upon the advertising and scanning schedules used [13]. Consolidation may take place in phases creating partially consolidated records during these phases. The number of fully consolidated records (150) is an upper bound of possible contacts. The partially consolidated records that are uploaded could be more by an order of two, and it could be about 25 000 [13] for a typical subject for the entire incubation period of 15 days.

• Smartphone’s system clock is assumed to operate at 2 GHz. Server upload bandwidth is 2 Gigabits per second (Gbps). Every functional entity of the solution can have 2- or 4-Gbps bandwidth.

• Proximity data for the last 15 days (14 rounded off to 15) are uploaded to Server as per US CDC guidelines [1].

There are other data points that are used in this analysis that are taken from current initiatives. For instance, the message sizes used in various computations are from respective initiatives. The sources of these data points are identified at appropriate sections in this paper. The CPU cycles used for various computations are estimates based on various Internet sources that are not published work. The resource requirements for the following operations are evaluated with the above data:

1. Generating unique UserID.
2. Server Network bandwidth requirement for TempID distribution.
3. Processing Cycles required to construct TempID from UserID and other parameters.
4. Server Network bandwidth requirement for Proximity Data Upload.
5. Server Network bandwidth requirement for follow up action download.
6. Storage requirement for the uploaded records.
7. Time required to map UserID to user mobile ID.

1. Generating Unique UserID: Generating unique UserID is part of the registration phase. A random 21-byte UserID is generated and then checked for uniqueness using a procedure such as Bloom filter [14]. Exhaustive checking is not efficient. There are other methods to generate unique large random numbers with \(O(1)\) complexity. The other option is creating a UserID with a time component so that duplication is not possible.

2. Network bandwidth requirement for PID distribution: Every registered user of 275 million should be sent one TempID of size 150 bytes (payload size of 84 bytes and packet overhead—rounded off to 150 x 8 bits) every 15 minutes. This transfer requires the server bandwidth of

\[275 \times 10^6 \times 150 \times 8/15 \times 60 \text{ bps} = 366\,666\,667 \text{ bps} (i.e. 0.367 < 2 \text{ Gbps}).\]

This value is less than the bandwidth of the server. This analysis is based on a simple push model. That is, each App is automatically sent a TempID every 15 minutes. A simple secure pull model will require nearly 10 times the size of this bandwidth. That is, an App needs to send a request for a TempID, then it will be sent one or more in response. A negotiated exchange over a secure transport layer such as TLS or DTLS will require about 10 times the bandwidth of the Push model. The 4 Gbps dedicated bandwidth of the server is sufficient for any of these options for the assumed load.

3. Processing Cycles required to construct TempId: The processing resource required to construct a TempID from PID and other parameters is computed here: It takes about 1000 cycles to prepare 100 bytes (84 bytes rounded off to 100) of AES-GCM authenticated data. This means that the CPU cycles needed per second for generating 275 million TempID over a period of 15 minutes

\[275 \times 10^6 \times 1000 \text{ cycles} / 15 \times 60 = 305 \text{ million CPU cycles/second}.\]

A dedicated server or a server with an encryption accelerator can host this for the above load.

4. Network bandwidth requirement for Proximity Data Upload: Assuming there are close to 6500 (75 000/12 rounded to nearest 500) people getting infected every 2 hours, this requires 6500 uploads every 2 hours. A person collects on the average around 150 fully consolidated proximity records on a typical day. The partially consolidated records that are uploaded are about 25 000 [13] for a typical case for a day. The size of each record is around 100 bytes. The upload of these proximity records must be done within the 2-hour period. The required transfer rate is equal to:

\[6500 \times 25\,000 \times 100 \times 8\text{ bits} / 2 \times 60 \times 60\text{ seconds} = 0.01805 \text{ Gbps} (< 1 \text{ Gbps}).\]

5. Network bandwidth requirement for follow-up action download: There is a follow-up action, each contact is sent a message (approximately 600 bytes in size to user App including overhead—an SMS text message) sug-
gesting that the contact will be hearing from healthcare officials regarding their contact status. This is like the last computation in item 4 with a different message size. This requires the worst-case bandwidth of assuming 150 (upper bound) contacts for every day for each case

$$6500 \times 150 \times 15 \times 600 \times 8 \text{ bits} / 2 \times 60 \times 60 \text{ seconds} = 9,750,000 \text{ bps} (600 \text{ Gbps}).$$

6. **Storage requirement for the uploaded records:** The storage required for a single day at the data centre for the uploaded records is computed here. There are 75,000 new cases and each case with 25,000 partially consolidated records with each record of size of 100 bytes. The storage could be reused.

$$75,000 \times 25,000 \times 100 \text{ bytes per day} = 187 \text{ Gigabytes/day}.$$

7. **Time required to Map UserID to user Mobile Number:** There are CPU cycles required to decrypt the TempID and find the mapping record. The decryption of TempID may not be an issue; however, the mapping could be. The Random access for a record with no structure to its key is I/O intensive and could consume time in the order of a few milliseconds and more than 2 hours for average values of 6500 new cases and 2250 (15 \times 150) records per case as illustrated below. An interworking solution may need to work with a larger population. The recent Covid data show larger average values for new cases. Thus, there is a compulsion to enable faster random access. This means that the PID should have an additional property: PID should support the necessary encoding that enables operational efficiency without compromising the anonymity property. Accomplishing this without compromising anonymity is going to be an interesting problem. This requirement may not likely be necessary for some distributed designs [5–7].

$$6500 \times 2250 \times 5/1000 \times 60 \times 60 \text{ hours} = 20.6 \text{ hours}! (>2 \text{ hours}).$$

The 5 ms is based on disk I/O time for a single retrieval, which is an optimistic average value. The number of proximity records of 150 for a day is an upper bound value. There is a certain amount of non-determinism here that does not favor the resource requirement of mapping operation. The average access time can fluctuate widely with number of cases, data distribution in disk, access methods, page fault, etc. Unless addressed, this I/O time issue could prevent the use of this centralized design for countries with large populations. This is a critical issue that needs addressing. Solid-State Drive could offer an improvement nearly by an order, but the other factors that could still likely push this value to more than 2 hours.

The complexities of all the operations are the linear function of the number of new cases per batch except for the RPID distribution operation. The RPID distribution can also be managed with the assumed server capacity. The mapping operation’s random-access time, however, could still prevent the use of this centralized design for a global solution. A solution is needed to address this large I/O access time; otherwise, the decodable property of this design becomes questionable. The review of a centralized Contact Tracing design concludes with this finding. The next Section provides a similar review for the distributed design with Exposure Notification, a Google and Apple initiative.

### 5. THE FOUR OPERATIONAL PHASES OF EXPOSURE NOTIFICATION

The registration operation downloads and installs the App on the subject’s smartphone. The Server registers the user’s participation by storing the user’s mobile number in a database. This database content may be accessed sequentially later for other operations. A considerable part of the rest of the operations is performed in the user’s smartphone in this distributed solution.

#### 5.1. PID management phase

When an Exposure Notification App is installed on a smartphone, it takes runtime permission for the resources in the phone. Once this is done, then a 16-byte random number, Temporary Exposure Key (TEK), is generated using Cryptographic Random Number Generator on the phone. The TEK is generated from a large domain, but still there is a miniscule probability that two or more TEKs generated on a day by different Apps are the same. A Rotating Proximity Identifier (RPID) Key is derived from TEK using Hashed Key Defining Function (HKDF). The RPI Key and Discretized Unix Time are then used to derive an RPID. The RPI is the RPID of Exposure Notification. An RPI is generated every 10 minutes. A generated RPI is used in PD exchange for about 15 minutes. Note, some of the RPIs may not be used in exchange. This is repeated for 24 hours, and then a new TEK is generated. A TEK and its generation time T together form a diagnostic key. The relationship among TEK, RPI Key and RPI is illustrated in Fig. 3. The TEK and RPI are two types of PIDs used for different roles, and both are rotated over time. The only way to check if a given RPI belongs to a smartphone is by requesting every registered smartphone.

#### 5.2. Proximity data exchange phase

The user anonymity is maintained by using RPI as PID. The proximity message ADV_IND is a BLE advertisement with
RPI and other proximity data (Fig. 4) in its 31-byte payload. The proximity information is exchanged by broadcasting proximity messages and scanning the advertisement broadcasts of the other smartphones. The RPI can only be mapped with the TEK from the smartphone that generated it. The RPI is not bound to any hardware address or fixed logical address on the smartphone. The proximity data exchanged includes RPI, and the data needed to compute BLE signal strength at the source. The timestamp and the signal strength at the receiving end are added to the record before storing it in the log. The ADV_IND can carry different types of payloads besides proximity payload. The Service UUID is used to uniquely identify proximity payload.

This processing is distributed to every smartphone that is not bound to a case. The matched data are correlated, consolidated and assessed for their proximity and duration values. An appropriate recommendation is made to the App user if the values meet the CDC guidelines.

5.4. Evaluating the PID of exposure notification

The operational description from the earlier part of this Section provides a context to evaluate the Exposure Notification’s PID.

1. **Unambiguous:** The TEK is chosen randomly from a large pool of numbers $\approx 10^{39}$. The RPID is a function of TEK and Discretized Unix Time. There is a miniscule possibility that two RPIs are the same on the same day in two or more Apps. This duplicate RPI could result in *False Positive* incidence. Considering the extremely low probability, and the known implication, RPI can be considered *Unambiguous*.

2. **Domain Size:** The size of the recycled RPI pool is $10^{39}$. This is more than enough to cover the world population with multiple RPI assignments to each subject. There is also a time component to qualify an RPI.

3. **Anonymous:** There is nothing in the RPI that is indicative of the subject’s identity.
4. **Non-trackable:** The RPI is rotated approximately every 15 minutes making it difficult to do *Wireless tracking* beyond 15 minutes.

5. **Robust:** It is not easy to copy or guess the TEKs used in a smartphone.

6. **Constraint-Free:** The TEK and RPI can be generated and exchanged by an active smartphone. This is not contingent on Internet connection. The proximity data payload is small enough to support both intrusive and non-intrusive data exchange methods.

7. **Decodable:** When a new case $Y$ is identified, there are two options to find all the contacts of this new case. The first option is intuitively simple but not viable. Each smartphone sends its entire log to the smartphone of $Y$ for processing. This requires every smartphone to have a large server-like capacity. The bandwidth and the computing capacity required are a function of the product of the number of registered smartphones and the average size of the log file. This option also changes the design to centralized from distributed. No other computing entity other than $Y$ can do this checking. The chosen anonymity property of RPI is the root cause of this *device affinity*. The only viable second option distributes the operation to all the registered smartphones. Every smartphone $X$ fetches all the 15 diagnostic keys of $Y$ from a server and checks if any of the proximity records in its log is generated by $Y$. This checking can be exempted for the smartphones of new cases. This option introduces its own issues. It puts every smartphone in the operational cycle of every new case and thus creates an enormous overhead. It also creates a large bandwidth requirement and other issues as illustrated in the next section.

5.5. **Affordable resource requirement—validation**

The following resource requirements are estimated for the validation of *Affordable Resource* property:

1. Server bandwidth requirement for Upload.
2. Server bandwidth requirement for Download.
3. Processing time required in each smartphone to check if the user is a *contact*.

**Server bandwidth requirement for Upload:** The bandwidth requirement for the server to upload 15-days *diagnostic* keys for every one of the 6500 cases in 2 hours is computed here. The size of a single *diagnostic* key is 20 bytes (16 bytes of TEK and 4 bytes of *Time*). A single packet with an overhead of about 60-bytes is sufficient to send all the 15 keys for a case. This bandwidth of 2.5 Kbps is insignificant even when the handshake overhead is added to it. The server bandwidth required for this

$$6500 \times (15 \times 20 + 60) \times 8 \text{ bits/2 hours} = 6500 \times 360 \times 8 \text{ bits/2 hours} \approx 2.5 \text{ Kbps} (<< 1 \text{ Gbps}).$$

**Server bandwidth requirement for Download:** The bandwidth requirement for the server to download 15-days *diagnostic* keys of new 6500 cases in 2 hours to every other registered smartphone is 695 Gbps. A single packet with an overhead of 60 bytes to send all the 15 keys for a case is assumed. Unlike centralized design, this periodic download is a function of the product of number of new cases and number of registered users. In the peak of the pandemic, this bandwidth requirement could become much larger. Multiple cases could be sent in one packet but that could create delay in finding a *contact*.

$$6500 \times 350 \times (275 \times 10^{-4} - 6500) \times 8 \text{ bits/2 hours} \approx 695 \text{ Gbps}.$$  

The specification [5–7] suggests smartphone-solicited (fetched) download. The solicited download could require considerably larger bandwidth with handshake overhead. The required bandwidth could also have a large variance around its average value as the pandemic goes through its multiple waves. These observations suggest that the distribution operation even with an optimal exchange method requires large, gracefully scalable bandwidth. A distribution hierarchy could be built to distribute the diagnostic keys. The Cell Broadcast Service [15] of cellular networks is a possible option. However, its use is restricted to emergency services. Another potential solution is distributing the keys using a registered configuration intensive IP multicast group. There is no discussion of any of these in the design specification [5–7]. This is likely due to lack of feasibility and other issues. Besides network bandwidth, the processing operation periodically requires considerable CPU cycles in every smartphone as illustrated below. There are many occurrences where the average number of news cases for a 2-hour period exceeded 6500 by about 4-fold.

**Processing time required to find contact:** In distributed *Exposure Notification* design, there are three steps to finding the matching RPID:

- **Generating RPI:** An RPI Key is generated from the TEK using a *Hashed Key Definition Function* (HKDF). Using this key and *Advanced Encryption Standard* (AES), an RPI is generated for every 10-minutes interval. There are 144 10-minutes intervals per day. This requires about 1500 CPU cycles for each RPI, ~1000 CPU cycles for HKDF and ~500 CPU cycles for AES.

Every 2 hours, the diagnostic keys of 6500 new cases are fetched by every smartphone from the cloud server for post-processing. There are 15 diagnostic keys for every new case. The number of CPU cycles required to generate RPIs is

$$6500 \times 15 \times 144 \times 1500 = 21 \text{ 060 000 000 cycles}.$$  

This is about 11 seconds of CPU time with a 2-GHz clock.

1. **Time required for RPI matching operation:** Partially consolidated records in the phone’s log (for 15-minute
interval) are examined for a match with each RPI. This lookup time is assumed to be about 100 CPU cycles per proximity record on the average. The specification [5–7] suggests that matching intervals should be longer than 15 minutes to cover the delay between generation and distribution of RPI. Matching time includes one 15-minute interval before the target interval and one after. The partially consolidated records per interval of 15 minutes are 25 000/(15 × 144). The comparison operation is performed for each RPI generated for each new case.

The total number of clock cycles needed to complete the processing for 6500 new cases is 6500 × 15 × 144 × 100 × 3 × 25 000/(15 × 144) CPU cycles. This is about 24 CPU seconds with a 2-GHz clock, and it is expected to vary with the number of partially consolidated records. The total processing time is 35 CPU seconds (step 1 and step 2). The 35 seconds of CPU time is significant for a smartphone, and this could interfere with the normal operation. This operation needs to be repeated every 2 hours in all the 275 million registered smartphones.

Occasionally, a phone may lose connectivity to the server for an extended period, typically a few hours. This can happen due to lack of battery charge, absence of the service due to billing issues, lack of connectivity, intentionally turned off smartphone, etc. This creates the risk of missing out contact identification opportunities during these intervals. These interruptions could force the smartphones to fetch and process diagnostic keys for multiple periods or make the smartphone skip over diagnostic keys of multiple periods. The former option could overload the phone and the server. In the latter option, there is a risk of not identifying a contact.

1. Additional processing with matching records: If there is a match, then the proximity distance, and duration are computed. This operation is significantly smaller in comparison to the other two.

This concludes the PID analysis of distributed Contact Tracing design with Exposure Notification. The following Section compiles the properties that are revealed in the last two Sections.

6. UPDATED PROPERTY LIST OF PID

Section 2 presented the list of obvious properties of PID. The reviews of the PIDs of two distinct designs in the latter two Sections 4 and 5 revealed four more properties. All these four properties are described in this section.

1. **Encodable/Decodable property**: PID encoding and decoding should be operationally efficient without compromising the other properties, specifically the anonymity property. Procedural encoding and decoding that does not require storage access can improve operational efficiency. This is illustrated by the PID mapping problem of centralized designs. The need for the next two properties is observed in the evaluation of distributed designs. They are applicable for the PIDs of both centralized and distributed designs.

2. **Non-affinity property**: The first of the two properties suggests that the PID design should not constrain any of the operations to a specific device. The resolution to such constraints could be expensive in terms of resource requirements. In distributed design [5–7], only the smartphone that generated an RPI can confirm the validity and the source of the RPI. The realization of anonymity property is the root cause of this affinity issue. The design works around this affinity issue with an expensive alternative as illustrated in Section 5. The choice of similar anonymity realization causes an I/O bottleneck problem in centralized designs [4].

3. **Low overhead property**: This property suggests that the computational overhead should not be unreasonably high. The distributed design [5–7] incurs high overhead in processing proximity data. Every phone needs to mimic the key generation procedure for all the 15 diagnostic keys of every new case. The probability of such computation identifying a contact is low, still the operation needs to be performed in every smartphone incurring enormous network bandwidth. The root cause of this is again the anonymity realization. There can be other reasons of such large overhead.

4. **Routable property**: In certain distributed designs [9], the PID encoding should include route data. These data are used to route the proximity records to the right server for post-processing operation.

7. CONCLUSION

This paper explored the properties and management of PID with the objective to develop an interworking global Contact Tracing solution. The PIDs of two Contact Tracing solutions, one with centralized and another one with distributed design, are studied to evaluate the implications of PID properties and PID management on Contact Tracing solutions. This exercise revealed that centralized design could run into I/O bottlenecks and distributed solutions could require large network bandwidth for their respective operations. Unless these issues are addressed neither one could be suitable for a global solution. This analysis also helped us to identify some non-obvious critical properties of PID. The properties presented in this paper are used in building a distributed global Contact Tracing design [9].

The data used in this paper are accessed from the following sources:

- Worldometer Coronavirus ([https://www.worldometers.info/coronavirus/country/us/](https://www.worldometers.info/coronavirus/country/us/))
- Statista ([https://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us/](https://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us/))
- The two designs cited in this paper—TraceTogether [4] and Exposure Notification [5–7].

---

**Section B: Computer and Communications Networks and Systems**

*The Computer Journal, Vol. 00 No. 00, 2021*
SUPPLEMENTARY MATERIAL

Supplementary material is available at www.comjnl.oxfordjournals.org.

REFERENCES

[1] Public Health Guidance for Community-Related Exposure. (2020) COVID-19 Guidance, Centers For Disease Control and Prevention, US. https://www.cdc.gov/coronavirus/2019-ncov/php/public-health-recommendations.html

[2] Martin, T., Karopoulos, G., Hernández-Ramos, J.L., Kambourakis, G. and Nai Fovino, I. (2020) Demystifying COVID-19 digital contact tracing: a survey on frameworks and mobile apps. Wirel. Commun. Mob. Comput., 2020, 1, 8851429–29. doi: 10.1155/2020/8851429.

[3] Contact Tracing App – India. (2020) Arogya Setu Official site: https://www.mygov.in/aarogya-setu-app/

[4] Bay, J. et al. (2020) BlueTrace: A privacy-preserving protocol for community-driven Contact Tracing across borders, A White paper from Government Technology Agency, Singapore https://TraceTogether.io/static/TraceTogether_whitepaper-938063656596c104632def383eb33b3c.pdf

[5] Exposure Notification API Preliminary 1.2. (2020) Apple & Google joint Contact Tracing Initiative 2020, https://developers.google.com/android/exposure-notifications/exposure-notifications-api

[6] Exposure Notification Bluetooth Specification 1.2. (2020) Apple & Google joint Contact Tracing Initiative, https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-BluetoothSpecificationv1.2.pdf

[7] Exposure Notification Cryptography Specification preliminary 1.2. (2020) Apple & Google joint Contact Tracing Initiative, https://blog.google/documents/69/Exposure_Notification__Cryptography_Specification_v1.2.1.pdf

[8] Bluetooth Core Specification v5.2. (2019) Bluetooth Core Specification Working Group, Bluetooth SIG Proprietary, https://www.bluetooth.com/specifications/archived-specifications

[9] Narayanan, H.T.S. (2021) Novel distributed and federated contact tracing systems – a global solution. TechRxiv https://www.techrxiv.org/articles/preprint/Novel_Distributed_and_Federated_Contact_Tracing_Systems_A_Global_Solution/14661138.

[10] nixCraft. (2009) Linux Tips, Hacks, Tutorials, And Ideas In Blog Format Humor: On Feb, Friday 13, 2009 Unix time Will Be 1234567890, Cyberciti.biz, https://www.cyberciti.biz/tips/february-13th-unix-time-will-reach-1234567890.html.

[11] Worldometer on Corona Stats. (2020) A Web portal that reports COVID-19 Stats and other useful information: https://www.worldometers.info/coronavirus/country/us/.

[12] O’Dea, S. (2020) Smartphone users in the United States 2018–2024, an online research report published in Statista: https://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us.

[13] Narayanan, H.T.S. (2021) Scheduling proximity data exchange for contact tracing. TechRxiv https://www.techrxiv.org/articles/preprint/Scheduling_Proximity_Data_Exchange_for_Contact_Tracing/14730648.

[14] Bloom, B.H. (1970) Space/time trade-offs in hash coding with allowable errors. Commun. ACM, 13, 422–426. doi: 10.1145/362686.362692.

[15] 3GPP TS 23.041 V2.0.0. (1999–10) Technical realization of Cell Broadcast Service (CBS), 3rd Generation Partnership Project; Technical Specification Group Terminals. https://www.3gpp.org/specifications