Novel Distributed and Federated Contact Tracing Systems – A Global Solution

Hari T.S. Narayanan, Senior Consultant

NetTools Consulting, ts.hari@gmail.com, Mobile: +91 9841289525

Abstract – There are several contact tracing solutions, some are for closed user groups, some are for the residents of a country, and some are open solutions with no clear boundary defined. These independent solutions are not adequate to support pre-pandemic global mobility. Ideally, what is needed is a global solution that can support decentralized control over data and at the same time support proximity data exchange among Apps developed by different vendors and for different countries. This paper proposes a family of contact tracing designs with *low-risk anonymity* that includes a centralized design, a distributed design, and a federated design for global solution.

Keywords - Contact Tracing; Interworking; Bluetooth Migration; Federated, Distributed, Centralized designs; SARS-2 COVID-19; Proximity Data Exchange; Exchange Schedule; Post Processing of data; Smartphones; Apps.

1. INTRODUCTION

A contact tracing system collects, stores, and computes data to identify the *contacts* and the *cases* for a contagious disease. A subject is a *case* when the subject is having infection specific symptoms or signs. A subject is a *contact* if the subject has contracted the virus but likely asymptomatic. The health care authority in a country develops the necessary guidelines which define the terms *contact* and *case* for an infection. The guidelines developed for an infection by a country can differ from the other ones in subtle ways. In this paper, the guidelines released by the Centre for Disease Control (CDC) and Prevention, US are assumed. The CDC guidelines suggest that if a subject spends more than 15 minutes within 6 feet of SARS-2 COVID-19 infected person, then the subject is a *contact* and a potential *case*. A *contact* can turn into a *case* within 14 days of becoming a *contact* and not likely after this period. The complete guidelines with all the criterions can be accessed from CDC's official site [1].

There are several contact tracing solutions [2], some are for closed user groups [3], some are for the residents of a country [4, 5], and some are open solutions with no clear boundary defined [6-8]. All currently deployed contact tracing solutions [2-8] are built mainly with smartphones. The Bluetooth LE (BLE) in smartphones is used in exchanging proximity data. The distance between two BLE devices is computed as a function of differential signal strength between BLE transmitter and receiver. The duration is computed by consolidating and correlating the periodically collected proximity data. BLE devices are built to find each other unlike other wireless technologies like WiFi and Cellular. A BLE device can be configured with a schedule to advertise its presence and to scan for other BLE advertisements within its operating range. The only modification required is the definition of a new BLE

service payload for proximity data exchange; Bluetooth supports such modifications.

There are two approaches to post processing of proximity data in current literature. In one approach, the proximity data of a new *case* is uploaded and processed in a *centralized* server [4,5] to identify the *contacts*. In the second approach the data is processed in every participating smartphone, thus *distributing* the processing load [6-8]. The centralized designs[5] may fail to scale due to random access required in handling large numbers of new cases, and other reasons. The single key solution that protects the proximity data could make the entire system vulnerable when the key is exposed. The distributed design[6-8] includes smartphones in the operational loop. This could create large operational overhead, resource requirements, and unreliable results.

There are two methods for exchanging proximity data; it is done either in an *intrusive* or *non-intrusive* way. In the *intrusive* methods [5], a smartphone opens a BLE connection to another smartphone after finding it and gets the proximity data exchanged. In the *non-intrusive* methods[6-8], the data is scanned and extracted from the payload of periodic advertisement broadcasts sent by BLEs; no one-to-one connection is required. The one-to-one connection to every other BLE requires more power and time. This limits its scalability and reliability of data exchange. The *non-intrusive* method on the other hand scales better with broadcast-scanning combination; and it is more secure. However, it is limited by its 31-byte payload when using Bluetooth 4.1 (BLE), the version with the largest user base.

Most of the contact tracing solutions use two types of identifier structures; a proximity identifier (PID) to track user's participation in contact tracing, and a set of rotating proximity identifiers (RPIDs) for operations. The solution can correlate the RPIDs of a user to the

user's PID when required. Each RPID is required to identify one of the users uniquely and anonymously. All major operations of contact tracing are built around these two identifiers.

The *proximity distance* is measured as a function of the differential BLE signal strength between the observer's (E_o 's) and observed entity's (E 's) smartphones. For privacy reasons, RPID of the observed entity E is made *anonymous* to the observer. That is, the RPID cannot easily be mapped to corresponding smartphone or user identity. The location data also infringes on the privacy laws in certain countries; thus, only the distance of E from E_o is recorded. For privacy reasons, the anonymity supported by a contact tracing solution is classified into three categories: *no-risk* or *low-risk* or *non-anonymous* anonymity.

In *no-risk anonymity*, the RPID is a random value that uniquely identifies the smartphone user anonymously. It reveals the user identity E only under *controlled* conditions. In centralized designs[4,5] the mapping between RPID and the user is maintained in a server. Only this server can map RPID to E . In some distributed designs[6-8], this mapping is possible only in the smartphone that generated the RPID. Thus, RPID needs to be sent to every participating smartphone to find the mapping. In *low-risk anonymity*, the identifier is an encrypted value of E . This is only as strong as the cipher and the mapping function used. Anyone who knows the cipher and mapping function can identify the corresponding user and user's location with a certain accuracy. The encryption method and the mapping function are not shared over the network. They are rotated periodically and across users to make this anonymity more robust. Given sufficient time and resources one can break the cipher and mapping function using plain text attack or exploiting the other vulnerabilities present. This means, one can possibly find out the past locations visited by an entity. The *no-risk* anonymity is safer than *low-risk* anonymity, however, poses certain challenges. In centralized designs[5] it creates a bottleneck with random access to thousands of records; in distributed designs [6-8] it creates large resource requirements, enormous overhead, and unreliable results due to data loss.

A contact tracing solution includes an operational component (24x7x365) with a large user base. There are close to 3.8 billion smartphone users in the world [9]; and this number is growing at 10% every year for the last few years. The solution should scale gracefully to these target values. A global solution requires cooperation among all the countries of the world. The exchange of data should happen without jeopardizing the privacy control of a country over its residents'

proximity data. The solution should scale with the user base, support country specific privacy solutions and post processing logic. The membership management procedures should be simple and agile; joining and leaving the global solution should have minimum overhead. The solution should be operationally viable and evolve to accommodate newer technologies and contact tracing for newer epidemics and pandemics. It should support all the critical use cases that are likely, such as large gatherings and crowded transit points. The current solutions do not adequately address these requirements. In this paper we describe a design that does.

Section 2 describes our proposed centralized design along with its operation and evaluation. Section 3 presents a distributed design evolved from our centralized design and its evaluation. This distributed design is different from the current ones [6-8]. The processing is distributed based on the available capacity and problem size instead of distributing it to every smartphone. Section 4 describes a federated design evolved from the distributed design along with its operation and evaluation. The coexistence of BLE and Bluetooth 5.0 is discussed in section 5. Section 6 presents a short summary of the paper.

2. NETTOOLS' CENTRALIZED CONTACT TRACER

Contact tracing is an operational system. There are procedures that are performed periodically round the clock. A contact tracing system includes the following four phases. In general, user registration is performed once. The other three phases are repetitive.

1. User Registration.
2. Identifier Management: Periodic creation and distribution of RPIDs.
3. Proximity or Contact data Exchange.
4. Proximity data upload and download for Post Processing.

The operation cycle starts for a user when this user downloads and registers contact tracing App. The operation creates an entry for the user in the registration database with MSISDN as key. The entry indicates the user's participation in contact tracing. Rest of the section describes NetTools' Centralized Contact Tracer (NC-CT) design and its evaluation.

The unit of temporal operational data of a user is referred to as Proximity Record (PR); and Proximity Data (PD) is a set of PRs of one or more users. A PR associated with a user is created at a server group and it may be consumed at the end within another server group after it goes through the transformations shown in Figure 1. There is sufficient information at the

receiving server group to map the transformed PR to its original values without resorting to storage access. The operational data starts as Proximity Record 1 (PR1) in the Identifier Management phase, then transforms into Proximity Record 2 (PR2) in Proximity Data Exchange phase, and finally into PR3 during Post Processing phase.

The user's MSISDN phone number is used as Proximity Identifier (PID). The MSISDN (15-digits) is represented in BCD form and then mapped using a mapping function (MF) to a 12-byte value, Operational Proximity Identifier (OPID). The mapping function hides known patterns in a Proximity ID. The proximity record PR1 is constructed at a server with the OPID and other data fields. The OPID, Dispatcher ID (D-ID), and Mapping Function ID (MF-ID) of PR1 are encrypted. This encrypted 16-byte component is the RPID. The PR1 is periodically sent to the App for proximity data exchange. Each App builds PR2 with the received PR1 and uses it for proximity data exchange. When a smartphone scans and finds a PR2 that is broadcast by another smartphone, it constructs a PR3 with it; and stores the PR3 in a log for possible post processing.

The Dispatcher ID field in RPID uniquely identifies a server group that needs to receive the PR3s of a new case for post processing operation. This might look redundant for centralized designs with only one server group; its usefulness will be justified in Sections 3 and 4. The complete encoding of the Dispatcher ID and its use are deferred to Section 4. The function used in mapping MSISDN to OPID is identified by the Mapping Function ID (MF-ID). This function could be a hashing function with unique mapping in either direction or an encryption function or a logical function or more importantly a sequence of these functions. The Key-ID field points to the key and its parameters used in the encryption of the 16-bytes preceding it. The encryption used could be a standard or proprietary block cipher encryption[10]. Carrying only pointer values provides the necessary privacy and at the same time helps to retrieve the original content when the PR3s of a case is uploaded to the designated server group for processing. Encryption and mapping combination provides the necessary low-risk anonymity.

The App uses the RPID to exchange proximity data with BLE ADV_IND broadcast messages. The proximity data (PR2) exchanged includes the RPID, BLE Transmitted Power level, and other BLE data. All these data items are encoded using respective Length-Type-Value in BLE ADV_IND. A custom Service UUID needs to be defined with Bluetooth SIG for this application. The encoding of the 31-bytes of BLE

ADV_IND payload is illustrated in Appendix A. The proximity data (PR3s) uploaded to the server group for the post processing includes the above data, proximity interval, and proximity distance.

Fig. 1. OPID, RPID, PR1, PR2, and PR3

The functional components of NC-CT are shown in Figure 2. The Server Group contains the following entities. NC-CT requires only one server group.

1. **Registration Database:** This database stores the MSISDN of all the registered users. An entry for a user is created at the time of registration. This entry also includes other user data as permitted by the health care regulations in a country.
2. **Proximity Data Database:** The proximity data (PD) uploaded from the smartphone of a new case is stored in this database for post processing and downstream analysis.

Fig. 2. NetTools' Centralized Contact Tracer

3. **Tables:** There are three tables: Key table, Mapping Function (MF) table, and Dispatcher Address table. The key table contains the set of block cipher keys and their respective parameters used in constructing RPID. Each key is uniquely identified by a key identifier, Key-ID. The Mapping Function table contains all the mapping functions with their respective parameters. Each function in the table is uniquely identified by a mapping function identifier MF-ID. The Dispatcher Address table contains the addresses of dispatcher server(s). Each entry in the table is uniquely

identified by Dispatcher ID (D-ID). These tables are shared by the servers of the group.

4. **Registration Server:** This server handles App registration for contract tracing. The MSISDN number of the registered App is stored in the Registration Database. App is also sent relevant group information like server addresses and Dispatcher ID to bind the App to a specific server group and security information to communicate with the server group. The Dispatcher ID is the same as the group ID.
5. **RPID Manager:** RPID Manager is responsible for building and delivering PR1 to Apps periodically. The OPID along with Dispatcher Identifier and Mapping Function Identifier (MF-ID) are encrypted with a key. Instead of single PR1, multiple PR1s built with distinct keys and mapping functions can be delivered together. RPID Manager supports both solicited and unsolicited delivery.
6. **PD Dispatcher:** When a new case is identified, the PD log (PR3s) from the user's smartphone is uploaded to the currently configured PD Dispatcher. The PD Dispatcher stores the uploaded data in PD Database and decrypts the data for the consumption of PD Mapper. The key to use for decryption is found using the pointer in the received PR3 record. The PD Dispatcher function entity can be split into uploader and dispatcher functional entities if needed.
7. **PD Mapper:** Bulk of the post processing is performed by PD Mapper. The output of PD Dispatcher is the input for PD Mapper; the output of PD Mapper is the input for Status-Action Dispatcher. The post processing function first extracts MSISDN from each record. Then the records with the same MSISDN that are temporally closer to each other are consolidated and filtered. The filtered output is sent to Status-Action Dispatcher.
8. **Status-Action Dispatcher:** The Status-Action Dispatcher informs the user about the contact **Status** and **Action** to follow by posting a message to the registered App using SMS transport. The Action could be a recommendation for isolation or quarantine. This needs to be followed up by a health care worker. This is where the digital contact tracing transitions into a manual process.
9. **Semi Manual Uploader:** This uploader is used to enter PD into PD Database manually or semi-manually. The semi manual process provides a partial data recovery option when the phone of a case is lost [11]. The output of this is sent to PD Mapper for post processing operations.
10. **Mapping Function:** The mapping functions are maintained by the administrator in a persistent storage in a table. A mapping function with a set

of parameters maps the user's MSISDN to a 12-byte OPID. This function and its complementary function both have the same identifier. Their respective application context is used to differentiate one another. The mapping functions are rotated/recycled periodically as required. There is no valid time mentioned for an entry in any of these tables. It is left to the discretion of the App with some guidelines. All time units in all our designs are Unix Time. Each functional unit copies these functions ahead of time and hosts them in their respective buffers. These functions are recycled by the RPID Manager. The same function can be used beyond a cycle time if the receipt of periodic RPID fail.

11. **Key:** A key with a set of its parameters can encrypt and decrypt RPID. The description of each key is maintained in a persistent storage. Each server copies these keys ahead of time to their respective buffers. The keys are rotated (recycled) as required. The same key can be used for two non-consecutive cycles. The life cycle management of mapping function and key are explained in Appendix B. The parameters of the mapping function and ciphering key can be made transparent even to the administrator.
12. The *Data Analytics* and *Data Mining* applications are outside the scope of this design for now.

2.1 NT-CT Operation

The registration operation for NC-CT starts when a user installs the App on the user's smartphone. User completes the installation and starts the registration process with the Registration Server that is hardwired into the App. The server validates the App and the user, and then adds the MSISDN number to the Registration Database. The App receives addresses of RPID Manager, PD Dispatcher, group identifier, and security data. This security data is used exclusively for requesting RPIDs. The registration in general is an App initiated client-server, synchronous, reliable, secure, one-time interaction either realized using HTTP stack or Internet Stack.

An App can support multiple networking options like HTTP, Web socket, Internet and SMS. The first three use IP addresses and SMS use mobile addresses. HTTP supports synchronous App-initiated interactions, Web Socket supports synchronous peer to peer interactions, Internet(TCP/UDP) can support a variety of application specific interactions. SMS supports asynchronous peer-to-peer interactions. Some of these options are expensive or have constraints. For instance, broadcast to mobile phones is supported only for emergency applications. Multicast requires

expensive configuration and management effort; still may not guarantee complete coverage. SMS can support limited broadcast from the external SMS servers to message servers; this broadcast is not end to end. Higher packet loss and frequent connectivity loss are common to all these connections; full handshake for reliable exchange is necessary. Secure and HTTP client-server interactions include large handshake overhead besides HTTP encoding overhead. A contact tracing App or server can choose the appropriate type of interaction for an operation.

2.1.1 Identifier Management

The management of RPID is handled by RPID Manager. Every registered App gets a new RPID periodically along with RPID Manager's group identifier and server addresses. The RPID Manager is responsible for building and delivering PR1 to Apps. It supports both solicited and unsolicited delivery. The unsolicited delivery is SMS based and it is meant for smaller closed user groups. An App can periodically initiate a secure, reliable client-server connection to RPID Manager and request for PR1s. The RPID Manager chooses an encryption key and a mapping function. A PR1 is constructed with the chosen key and the mapping function and sent in response. The data piggy backed to PR1 is used to configure the server addresses and group ID at the App if required. Larger distribution loads can be supported with the same server bandwidth by batching multiple PR1s in each response and reducing request frequency. Solicited delivery of PR1 is secure, scales better, and more efficient than unsolicited.

2.1.2 Proximity Data Exchange

Once the RPID is received, the App starts exchanging the proximity data with the Apps running in other smartphones. The PR2 exchanged includes the RPID of the transmitting phone, Key ID, and the BLE Transmitted Power Level (Figure 3). At the receiving side, timestamp, and the received power level (Received Signal Strength Indicator – RSSI) are added to proximity data before it is stored in the log. Later, the App does a partial consolidation of this data for proximity duration and proximity distance. Each consolidated PR3 in the log contains RPID, Key ID, proximity interval, and duration (Figure 3). The log is uploaded to the server group for correlation and further consolidation when the user of the phone becomes a case.

2.1.3 Proximity Data Upload and Post Processing

If one of the subjects is infected, then the subject or the health care authority can upload the proximity data from the smartphone to the PD Dispatcher. This

uploaded data includes the PD log for the duration of the incubation period or longer. The dispatcher decrypts each PR3 in the uploaded log. After completing the decryption, the data is sent to the PD Mapper. The PD Mapper using the reverse mapping function extracts the MSISDN and consolidates the data for Status-Action Dispatcher. The Status-Action Dispatcher sends the Status and Action to the concerned subject's App over SMS after applying filtering logic. At this point, the manual process can take over in terms of counseling and follow up. Some of the later activities of contact tracing can also be automated using the same framework. One such activity is reminding the subject about his responsibilities during quarantine. Collecting the vitals of the subject is another such activity.

2.2 NC-CT Evaluation

The NC-CT design is evaluated based on how well it satisfies the following list of identifier properties and requirements. This list provides a common base to evaluate different contact tracing designs.

1. **Domain Size and Unambiguity:** The MSISDN is used for user identification. In general, each MSISDN maps uniquely to a single mobile subscriber. The size of MSISDN can be up to 15 digits. Since a smartphone is mandatory for this solution, the domain size of MSISDN is adequate to cover all the smartphone subscribers of the world.
2. **Anonymity:** MSISDN is not anonymous, but RPID is. The MSIDN to RPID mapping and RPID to MSISDN mapping are done using a pair of complementary functions and ciphering. The parameters used are random numbers and hosted securely. This mapped and encrypted RPID is used in PR1, PR2, and PR3. This mapping and encryption are done at the beginning of the operational sequence at the server, and decryption and reverse mapping are done at the end of the operational sequence at a designated server. The mapping functions and keys are secure and rotated often and across users to support *low-risk* anonymity.
3. **Non-Trackable:** The RPID is rotated periodically making it difficult to wirelessly track the subject beyond the chosen period.
4. **Decodable and Encodable:** Once a RPID is available, the corresponding PID (MSISDN) can be found procedurally using the key and the mapping function specified in the RPID. This procedure avoids expensive disk access. The procedure requires an inconsequential amount of CPU cycles. Supporting this property with low-risk anonymity eliminates the potential I/O bottlenecks of

centralized designs[5] and resource issues of distributed designs. Encoding complexity is quite similar.

5. **Operational Robustness:** There could be significant issues to the loss of smartphones[11]. Semi-manual partial recovery of PD is still possible here if the smartphone of a case is lost. The MSISDN is mapped and encrypted using rotating mapping functions and rotating keys. The function and encryption method used are known only at the server. Plain text attack possibility is mitigated by having sufficiently large numbers of keys and often rotating these keys. The single Key solution of [5] is generalized with multiple rotating keys. Keys can be replaced and upgraded without disrupting the contact tracing operation. Having multiple makes it easier to isolate and restrict the compromised data. If an App loses the connectivity to the RPID Manager, then the current RPID can be used until the next RPID is received without disrupting the service. A single RPID response message can carry multiple RPIDs to handle connectivity outages.
6. **Constraint and Limitations:** Some of the current centralized designs are limited to intrusive exchange methods due to their large BLE payload. There is no such restriction with NC-CT. The payload size is small, both intrusive and non-intrusive exchange methods could be used. This is one of the first solutions that permits the use of multiple proprietary and standard ciphering.
7. **Resource Requirement:** The bottleneck resulting from the random access seen in certain centralized designs and single key limitation are eliminated from the operational flow of NC-CT. There are no other significant differences to the processing phase of this and current centralized design[5].
8. **Resource Overhead:** Except for the differences mentioned in item 7, the design presented in [5] and NC-CT have similar RPID distribution and post processing phase. There are no resource overhead issues in both unlike the ones found in distributed designs[6-8].
9. **Non-Affinity:** The binding between an App and its server group is created by configuring the App with Dispatcher Group Identifier, and server addresses. This is a logical binding with domain addresses and logical identifiers, not to any specific server machine. This binding itself can be updated any time as described in the next section.

If a load balancer can distribute each registration request among n independent server groups, then there are n non-interworking contact tracing systems. Each one of these systems will fail to process the PR3 data from other groups. The next section describes a

distributed design that addresses this problem with n interworking NC-CTs.

3.NETTOOLS' DISTRIBUTED CONTACT TRACER

The NetTools' Distributed Contact Tracer (ND-CT) is a generalization of NC-CT, there can be multiple server groups in it (Figure 3). The changes required for the proximity data (PR1, PR2, PR3) is incremental. The Dispatcher ID field in a Proximity Record (PR) identifies a server group. The most significant byte of the Dispatcher ID field is 0, the least significant 7-bits encode a server group within the ND-CT that will be responsible for the Post Processing of the PD. There are incremental enhancements to server functions and no changes needed for App.

Each server group consists of the following enhanced server element instances: Registration Server, RPID Manager, PD Mapper, PD Dispatcher, Status-Action Dispatcher, and Semi-Manual uploaders. The tables are now served over a secure network by a server entity, Table Server. This server is common to all the server groups in the design. The data served by it are also common to all the server groups. This enables a PR1 to start from one server group and the corresponding PR3 to arrive at a different server group and still get processed. The PR3 can also be forwarded to its home group if required. If there are no privacy restrictions, a PR3 can be processed anywhere, otherwise it is forwarded to its home server group.

Fig. 3. NetTools' Distributed Contact Tracer (ND-CT)

The servers in each group include additional client functionality to request the entire table or one of the entries of a table. The Table Server can also send notifications to alert the servers about the status change of the table entries.. The PD Dispatcher is enhanced to include the function required to forward a PR3 to a PD Dispatcher in another server group. All these new tasks make use of the secure network that connects the Table Server, and various server groups. The dispatcher decrypts each PR3 in the log and checks if the value in the Dispatcher ID in the record matches its own. If there is a match the deciphered record is sent to the PD Mapper in the group for further

processing. If there is no match, then the ciphered record is dispatched to the PD Dispatcher specified in the record. The matching operation is explained in Section 4. There are no changes to the other core operations. Each server group includes its own Registration and PD databases. Inter group traffic is an added resource requirement besides the traffic to and from Table Server. This traffic mostly consists of PDs forwarded to their respective groups and interactions with Table Server. Since any server group can process any PR3, the PD traffic can be controlled. The other traffic is not part of the operational flow and it is sparse. The rest of the operations are identical to that of NC-CT, except for the new features that are supported to scale ND-CT.

Each server group behaves as an independent system. The cooperation starts after the log is uploaded and processed by a PD Dispatcher. Only the PD Dispatcher needs to be aware of the presence of the other groups or distribution. Any server group can process a PR3 and send the contact status and action to the App. Beyond this point, the manual process from the home group can take control. The NC-CT is a special type of ND-CT with a single server group.

3.1 ND-CT Evaluation

The core operations of ND-CT are no different from that of NC-CT. The PD forwarding and group management are ND-CT specific features. The following group management operations support scalability features with no disruptions to the ongoing operational sequence.

1. Scaling across with a new server group: A server group is created first and made active. The image of the App is updated to include the address of the Registration Server of this group as preferred server. The operation of the group starts when a new App is registered to the group. The existing server groups can also be configured to reject any registration request to redirect all new registration requests to this group.
2. Replacing the current group with a higher capacity Group: A new scaled-up server group is set up and activated. The image of the App is updated to include the address of the Registration Server of this group as preferred server. The content of the current registration database is copied to the new server group. The current Registration Server rejects all registration requests. The current RPID Manager updates every registered App with the new Dispatcher Identifier and server addresses by piggybacking them to PR1. When the new server is contacted for RPID by an App, the status of

corresponding MSISDN in the current group's database is set to Marked. After marking all the MSISDNs in the database, the current RPID Manager's address is aliased to the new RPID Manager and it is decommissioned. The aliasing can also be done when there are fewer unmarked MSISDNs in the current database. Once the decommissioning of RPID Manager is done, all the other servers are decommissioned after waiting for a full incubation period and a certain grace period. The grace period is to make sure the PR3s forwarded from the other server groups are not lost.

3. Scaling Across by splitting a single group into two: A set of MSISDNs that fit a certain pattern are copied to the new server group. The new server group is activated, and it is added to App image. The MSISDNs which match a required pattern are reconfigured to work with the new server group using PR1 distribution operation. When one of these Apps contacts the new server for RPID, the corresponding MSISDN entry is marked in the current database. The marked entries are removed after 2 weeks and a grace period.

A phone may get activated after a long dormant period. If the dormant period is more than a certain threshold value, then the contact tracing App of this phone is made to initiate an abbreviated special registration. This registration helps to initiate group transfers if required. Registration is not part of the operational cycle.

The marking and checking the database are the only new tasks that are added to the operational phase to support scaling operations, the rest are offline tasks. Even the marking and the checking tasks can be done in batches without disrupting the normal operations. There are no time constraints for these tasks. Thus, the random access to the database is not an issue. A smartphone may not have connectivity to servers for various reasons. Thus, it is essential to drive activities on events instead of time. For instance, the entry is marked only when the App connects to a new RPID Manager.

The ND-CT design is essentially evolved from NC-CT design. Thus, it retains some of the operational characteristics of NC-CT. Unlike a loosely put together NC-CT server-groups, the server groups of ND-CT are bound by the common table server. An ND-CT can process any PR at any server group. The Table Server is a lightweight entity decoupled from the timing requirements of main operational flow. The table service is offered over a secure VPN with a notification feature. The ND-CT supports a set of procedures that

can gracefully scale the distribution horizontally and vertically.

The solutions where processing is distributed to each smartphone require large network bandwidth to distribute PD data of every new case to all the registered smartphones. This requires a significant amount of smartphone resources periodically to complete the processing. The involvement of every smartphone in post processing operations of every new case creates enormous overhead. The unreliable connectivity to smartphones makes the results of post processing unreliable. The service disruption due to smartphone loss is more of a problem with these designs[11] than other designs. The ND-CT design addresses these issues by choosing larger loads for distribution. The ND-CT solution can be scaled to the size of the global population. It still falls short on privacy requirements needed for the global deployment. The next section describes NetTools' Federated Contact Tracer (NF-CT) design that addresses this issue and other global requirements.

4. NETTOOLS' FEDERATED CONTACT TRACER

The NetTools' Federated Contact Tracer (NF-CT) design (Figure 4) is evolved from NetTools Distributed Contact Tracer design. This federated design addresses the requirements of a global solution. An NF-CT includes multiple interworking ND-CTs in it. It is evolved from ND-CT with the following enhancements:

1. The Scope of Mapping Function service is local to respective ND-CTs. This protects the MSISDN only to an extent from non-home ND-CTs. This shortcoming is addressed at the end of this section.
2. The scope of the key server is global (federation). Keys are shared by all the ND-CTs in the federation. This is to enable the forwarding of PD from a one ND-CT to any other ND-CT in the federation.
3. There are two different PD Dispatcher Address tables. Global Dispatcher Address Table (GDAT) is NF-CT(global) scoped and the Local Dispatcher Address Table (LDAT) is ND-CT(local) scoped.
4. There is an enhancement to the encoding of the Dispatcher ID field to support the two different tables described in item 3.
5. The PD Dispatcher is enhanced – there is a PD Dispatcher Gateway to NF-CT in every ND-CT. This gateway also acts as a proxy server for an ND-CT in accessing the global key table.

The mapping functions table and LDAT are local to each ND-CT in a federation, and they are served over ND-CT's local secure reliable network. The key table and the GDAT on the other hand are served to PD Dispatcher Gateways of NF-CT over the secure,

reliable network that connects ND-CTs in an NF-CT. This network is also used for forwarding proximity data from one gateway in NF-CT to another. A PD Dispatcher Gateway is a specialized PD Dispatcher. The keys are common to all the participating ND-CT, thus cannot be proprietary. This arrangement necessitates the mapping function to include a ciphering component. This cipher could be proprietary or standard one. More on this at the end of this section.

The PD Dispatcher function and the encoding of Dispatcher ID are enhanced for NF-CT. The most significant 9-bits of the Dispatcher ID is used for inter-ND-CT forwarding and the value in the least significant 7-bits is used for intra-ND-CT forwarding. This allows 128 different dispatchers in each ND-CT and 512 global dispatchers. Every PD Dispatcher in the system is uniquely identified by the full 16-bit value. All the PD Dispatchers in an ND-CT share the same 9-bit global prefix. There is one PD Dispatcher Gateway in each ND-CT that acts as NF-CT gateway. The 7-bit suffix of the Dispatcher ID is 0 for all the gateways. All the traffic between ND-CT and NF-CT Network is directed through this gateway. The 9-bit prefix of Dispatcher ID is ND-CT identifier.

Fig. 4. NetTools' Federated Contact Tracer Design

A gateway uses the following forwarding logic when a PR3 is received: if the global component of Dispatcher ID in PR3 is different from its own, the PR3 is sent to the appropriate NF-CT gateway using GDAT. Otherwise, the PR3 is either processed locally or forwarded to PR3's home dispatcher using LDAT. Instead of forwarding record by record this can be accomplished by forwarding a batch of records. A non-gateway dispatcher uses the following logic when a PR3 is received: if the global component of Dispatcher ID is different from its own, the PR3 is sent to its NF-CT gateway. Otherwise, the PR3 is either processed locally or forwarded to PR3's home dispatcher. In either case LDAT is used to find the destination.

The mapping function is only meant to map MSISDN to OPID and vice versa. This mapping offers a weak privacy that is not adequate when a subject's PR is processed in a non-home ND-CT. The forwarding

procedure of NF-CT could be exploited to expose the MSISDN from OPID. This is not an issue with intra ND-CT processing, where the entire system is administered by a single entity. The following solutions are offered to address this issue.

An innovative mapping function could address the above issue; however, it may still not be adequate. One of the many proprietary or standard block cipher methods that operates with small sized data can be combined with mapping and logical functions. The Tiny Encryption Algorithm (TEA) [12,13] or one of its improved extensions XTEA or XXTEA[14] can be employed. There are several other block ciphering algorithms [15-16] besides TEA. These ciphering methods operate on 64-bit data blocks with 64-bit or 128-bit keys. They are simple to implement block ciphers, typically requiring a few lines of code. There are 12-bytes in OPID. Two blocks of 64-bits are created with 12-bytes of OPID by overlapping the middle word. First by encrypting the least significant 8-bytes, and then using the most significant 4-bytes of the result with the most significant 4-bytes of OPID. The decryption is done in the reverse order. Care must be taken to avoid those methods that do cyclical reversal. The other cleaner solution is using a cipher that operates with 32-bit blocks. The 32-bit block ciphers are less robust compared to 64-bit block ciphers. However, they still offer better protection than mapping functions. The encryption of the OPID, Dispatcher ID, and MF ID can make use of standard methods like AES, DES[17,18]. This approach preserves the same payload across all the three designs. More importantly, both exchange methods are still viable across all the three designs.

The NF-CT design offers decentralized control over proximity data and at the same time facilitates the forwarding of PDs among participating ND-CTs. The App and the PR2 exchange method are independent of this forwarding. The NF-CT design started with an objective to fulfill the requirements for a global solution. A country can join the federation and leave the federation at any time. This is supported by configuring the Dispatcher table, DNS entries, and enabling the access to global dispatcher and key tables. A PR is protected from the other ND-CT systems by rotating OPID ciphers and mapping functions. Partitioning of the Dispatcher table with global scope and local scope protects the intra ND-CT topology. Each ND-CT is behind its PD Dispatcher gateway, a single point of entry and exit; easy to administer for secure access. The ND-CT can scale gracefully using the procedures described in Section 3.1. The federation can scale with the support of Dispatcher Gateway and global key tables. The payload and other data structures used in

all the three designs are essentially the same. The payload is still small enough to support a non-intrusive exchange method.

There is no support for message digest in any of these designs. The next section describes an enhancement to address this and migration of these designs to Bluetooth 5.0. The other limitation is the vulnerability of these designs to plain text attack. Each user knows his MSISDN number, this information could be used in cracking the ciphering if the encryption method is known. The rotating key and unknown encryption method reduce the vulnerability from plain text attack. The access to RPID can be restricted only to the respective App. This is practiced in certain contact tracing solutions.

5. MIGRATING TO BLUETOOTH 5.0

The current contact tracing solutions are constrained to use BLE (Bluetooth 4.1) with its limited 31-byte payload because of its dominant installed base. This limits the size of the PR, and consequently PR needs to be sent without a message digest. The Bluetooth 5.0 is backward compatible with BLE and it offers a broadcast payload of up to 244 bytes. This feature can be used to encode PR with 2-levels of key protection, digest, and other enhancements. The enhancement should be less disruptive and support the coexistence of both for a seamless migration.

Fig. 5. CID, PR2, and PR3 of Bluetooth 5.0

An encoding for the new PR is shown in Figure 5. There is no change to Dispatcher ID in the new encoding. The input to mapping function is still MSISDN, however it is in ASCII form. This eliminates repeated ASCII-BCD transformations. The OPID is now 22-bytes, big enough to be encrypted with MF-ID using AES[17], DES [18] or TEA [12-14] or any proprietary block ciphering. The key used is identified by the Key-In field. The Key-in eliminates the workarounds described for 12-byte OPID in Section 4 and it is served by ND-CT scoped Server. There are no changes to MF-ID. The mapping functions need not include ciphering anymore. There are 4 unused bytes for future use. The 32-bytes with MF-ID are encrypted with the key specified in K-Out. This key is served by a NF-CT scoped server. Using this key, PD is decrypted and forwarded.

The RPID Manager can be enhanced to send legacy payload to BLE Apps and both payload types to

Bluetooth 5.0 Apps. Each Bluetooth 5.0 App sends two distinct advertisement messages, one with legacy BLE payload and another with new Bluetooth 5.0 payload. Each BLE App continues to advertise with legacy payload. A backward compatible Bluetooth 5.0 App can receive and process both payload types. There are incremental changes required to servers and database entities to support the coexistence of two different payload types. RPID Manager, PD Dispatcher, and PD Mapper are enhanced to support legacy and enhanced PRs. Scheduling of advertisement and scanning is an area of concern, sending two different types of advertisements from Bluetooth 5.0 adds more traffic to BLE advertising channels. This reduces the scanning reliability due to collision and reduced advertisement frequency.

6. CONCLUSION

This paper proposed a family of contact tracing solutions that includes a centralized design, a distributed design, and one of the first federated designs for global solution. These designs support low risk anonymity. This allows some major issues found in other designs to be addressed and provides a viable global solution with low-risk anonymity. The centralized design eliminates the random-access bottleneck found in comparable designs[4,5]. The support for multi-key solutions restricts the exposure of data when a key is compromised. The distributed design allows flexible distribution of processing load compared to other designs[6-8]. The distribution is based on the load size and available resource capacity. The solution supports graceful scaling features. The large resource requirements and large overhead issues found in other distributed designs[6-8] are avoided. The federated global solution allows individual systems to interwork and at the same time empowering each system to control and protect its data. The membership management is through configuration. All the three designs support multi-key management and the choice of intrusive and non-intrusive method of exchange. The App design, the post processing logic are independent of these designs. Thus, leaving much needed control to individual federation members and App developers. These designs can support the coexistence of Bluetooth LE (Bluetooth 4.1) and Bluetooth 5.0. This feature can ease the migration to Bluetooth 5.0.

REFERENCES

- [1] Public Health Guidance for Community-Related Exposure (2020), COVID-19 Guidance, Center For Disease Control and Prevention, US. <https://www.cdc.gov/coronavirus/2019-ncov/php/public-health-recommendations.html>
- [2] Tania Martin, Georgios Karopoulos, José L. Hernández-Ramos, Georgios Kambourakis,¹ and Igor Nai Fovino, (2020), Demystifying COVID-19 Digital Contact Tracing: A Survey on Frameworks and Mobile Apps, *Wireless Communications and Mobile Computing*, Hindawi, Volume 2020(October), Article ID 8851429, <https://doi.org/10.1155/2020/8851429>
- [3] PWC Automatic Contact Tracing: <https://www.pwc.com/us/en/products/check-in.html>
- [4] Contact Tracing App – India (2020), Arogya Setu Official site: <https://www.mygov.in/aarogya-setu-app/>
- [5] Jason Bay 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
- [6] Exposure Notification API Preliminary 1.2 (2020), Apple & Google joint Contact Tracing Initiative 2020, <https://developers.google.com/android/exposure-notifications/exposure-notifications-api>
- [7] 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/ExposureNotificationBluetoothSpecificationv1.2.pdf?1>
- [8] 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
- [9] O'Dea S, (2020), Number of Smartphone users Worldwide, an online research report published in Statista: <https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>.
- [10] The Citizens Compendium: https://en.citizendium.org/wiki/Block_cipher#Iterated_block_ciphers
- [11] Hari T.S. Narayanan (2020), Loss of Smartphones, and its Implications to Contact Tracing, *International Journal of Engineering Research and Technology (IJERT)*, Volume 9, Issue 7.
- [12] Wheeler, David J.; Needham, Roger M. (1994-12-16). TEA, a tiny encryption algorithm. *Lecture Notes in Computer*

- Science*. **1008**. Leuven, Belgium: Fast Software Encryption: Second International Workshop. pp. 363–366.
- [13] Matthew D. Russell (27 Feb 2004). "Tinyness: An Overview of TEA and Related Ciphers". Archived from the original on 12 August 2007.
  - [14] Roger M. Needham, David J. Wheeler (October 1997). Tea extensions.
  - [15] Treyfer, A Block Cipher/MAC Algorithm: <https://en.wikipedia.org/wiki/Treyfer>
  - [16] James L. Massey: SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm. Fast Software Encryption 1993: pp. 1-17
  - [17] Rivest, R. L. (1994). "The RC5 Encryption Algorithm" (PDF). Proceedings of the Second International Workshop on Fast Software Encryption (FSE) 1994. pp. 86–96
  - [18] "Announcing the ADVANCED ENCRYPTION STANDARD (AES)" (PDF). Federal Information Processing Standards Publication 197. United States National Institute of Standards and Technology (NIST). November 26, 2001.

Appendix A

Fig 6. BLE ADV_IND Payload Encoding with PR2

Appendix B

LIFE CYCLE MANAGEMENT OF MAPPING FUNCTION AND KEY

The life cycle management of keys and mapping functions are similar with minor differences. The differences are pertaining to the different operational servers which consume these tables. Instead of describing both, the life cycle management of mapping functions is described here. The MSISDN of a user is mapped to an OPID using a mapping function with a set of parameters. The table server supports both synchronous and asynchronous interfaces to operational servers over a reliable secure CT network.

Fig 7. Life Cycle State Management of Mapping Function

The mapping functions are common to all the operational servers of different server groups. They are maintained by the administrator in persistent storage in table form. The functions themselves are available as dynamically loadable modules. Each entry in the table includes a unique *ID*, *function Type*, and the function specific parameters to perform mapping and reverse mapping, a *Status* field, and a *Time field*. The status of a mapping function could be one of the values: {*Active*, *Retired*, *Deprecated*, *Obsolete*, *Null*}. When an operational server reboots it copies the entire table to its buffer. In general, it uses only those functions which are in *Active* state. The use of these states is illustrated by the state transition diagram in Figure 7. When the administrator creates a new entry, its status is set to *Active*, all the servers are sent a notification to update

their respective copies with this new entry. When the administrator decides to retire a function, the status is changed to *Retired* in persistent copy, and the time to retire is entered into the *Time* field of the entry. This time value is a function incubation period. All the servers are notified to update their respective copies. The RPID Managers will stop using this entry for mapping at once. The PD Mappers will continue to use the mapping function for downloaded PR3s until the status becomes *obsolete*. At the end of the incubation period, the status field of persistent copy is updated to *Deprecated*, and the time value is set to grace period. All the servers are again notified to update their respective copies. At the end of the grace period, the status is changed to *Obsolete* in the persistent copy and servers are notified. This entry can now have a new function, or the old function can be activated. All the servers are notified to update their respective copies when that happens.