Chapter 4
Outline of a Proposal and Conclusion

Abstract In this chapter, we discuss the future of digital contact tracing systems and the evolving role of cryptography in this regard. We mostly concentrate on how a decentralized system should be designed maintaining the privacy of the users. We first consider an ideal system and then move forward towards a more realistic one. The assumptions and cryptologic issues are explained. The system may also be used as a centralized one with certain design modifications. We discuss higher order contact tracing in terms of the neighborhood of a neighborhood, i.e., one more level of indirection. This is the concluding chapter of this brief manuscript.

4.1 Lessons from the Past

Over a hundred years ago when the world reeled under the Spanish flu pandemic, there were no smartphones or even mobile phones, but manual contact tracing was used as one of the measures to contain and control the spread of the infection. While there has been a tremendous advancement of technology from the days of Spanish Flu to the present days of COVID-19, including the re-emergence of Artificial Intelligence (AI) from its long hibernation to a state where experts declared that AI is the new electricity [1] or that the singularity is near [5], it failed to either predict or make humanity ready for the current catastrophe. So we must go back and take a look at the history of epidemics, understand the complex issues around the age-old epidemiological tools like contact tracing and develop realistic expectations of how technology can be leveraged to overcome some of these challenges.

In Chap. 1, while depicting the background motivation for adopting digital contact tracing systems, we have talked about the natural advantages of using digital tools over the manual contact tracing processes, like faster notification and intimation, better scalability, ease of manageability, etc. However, to build a holistic perspective,
we must learn the inherent complexities of the contact tracing process and re-evaluate the currently proposed digital automated solutions in light of that.

Dr. Eugenia Tognotti, in her historical review titled “Lessons from the history of Quarantine, from Plague to Influenza A” [6], has mentioned that, “Organized institutional responses to disease control began during the plague epidemic of 1347–1352.” She pointed out that even in 1830 when the first wave of cholera outbreak reached European ports, the health officials used more or less the same strategies as those used at the time of plague. In cities, the sick people were forced by the authorities to move into “lazarettos” or isolation hospitals and the people who came in contact with the sick people or traveled from a place of outbreak were quarantined, which means, the contact tracing process was actively used to identify such people. During this time, in some countries, special laws were passed that curbed personal freedom of individuals and many times such laws were misused by the authorities to suppress political opposition. The widespread usage of social measures like quarantine or isolation conflicted with an individual’s personal freedom and citizens’ rights.

While there could be concerns on possible excesses or misuse of power by the authorities by exploiting the fear of the disease in masses, one can also justify such measures to a “reasonable” degree with the “right intent” in view of safeguarding the public health and for ensuring the safety or greater good of the citizens. Herein lies the core challenge of contact tracing (irrespective of whether manual or automated), where it has to delicately balance the two opposing requirements. This has been captured nicely in a recent article [4] by Dr. Shelly Fan where she has said that “…contact tracing has always teetered on the line between individual freedom and the good of the general public; the stigmatization of a viral scarlet letter versus keeping others safe; the price of health data sharing versus societal responsibility.”

Let us understand the implication of the need where a DCTS needs to delicately balance the two opposite types of requirements. For example, on one hand, the system has to be prompt, accurate, and effective in alerting the general public of the risk of infection or exposure, while at the same time, it must not disclose the identity of a sick person. Additionally, in many countries, it may also allow a choice (or personal freedom) to that sick person on whether he/she is comfortable in sharing that status to the system. Similarly, the societal responsibility of minimal data sharing (e.g., the GDPR requirements in European countries) should not create a situation where it becomes tedious to carry out the usual epidemiological analysis of the contagion. Interestingly, this also means that a DCTS should be carefully architected in such a way that none of its features or strengths gets overused (e.g., anonymization of user data should not be done to such an extent that aggregate level analysis becomes infeasible). Otherwise, that feature or strength would become the biggest weakness or drawback of the system.

Now that we have touched upon the core challenges around the contact tracing process, let us explore a vision of an ideal version of the system before we settle down with a realistic one. As we go through the descriptions of these systems, it becomes apparent that, while a digital contact tracing system requires a broad-spectrum engineering focus (e.g., computing, networking, designing, usability, etc.),
at the core of it lies cryptology, which is the primary motivation for us to consider a cryptologic approach for this book.

4.2 An Ideal System

In this section, we allow ourselves to indulge in imagining a system (or more appropriately, a framework of systems) that may maximize the desirability factor of users. So we request the reader to temporarily suspend the other two perspectives—feasibility and viability for the time being and focus solely on a system that can be termed “ideal” from the user’s point of view.

The motivation for envisaging an ideal system comes from two fronts. First, it would give us the independence and freedom to bring innovative ideas and start the conceptualization process of digital contact tracing systems without getting constrained or influenced by the approaches already undertaken by the currently implemented (or proposed) protocols, APIs and apps. Second, it would act as a basis for us to compare a realistic framework and understand to what extent that framework may fall short of an ideal framework of the future.

The primary users of a DCTS are the individuals who at any instant of time, can be in different states of infection as per an epidemiological model, including (but not limited to)—susceptible or at risk or infected or diagnosed or recovered, etc. The secondary users would include the health authorities, the administrative authorities and the epidemiologists. There could be a third category of users who are interested to attack the system individually or as a collective, colluding group and may even attempt to influence a regular user (from the first two categories) to work in accordance with their directions temporarily or for an extended period of time. For this third category of users with malicious intent (or malicious users), the ideal system should not be desirable at all or, in other words, the system should appear to them completely unusable. We also assume that the first two categories of users always have positive intents and the only time they may behave with some negative intent (as rogue users) would be the time when they get somehow influenced by the malicious users.

Expected functionality from the perspective of primary users could be as follows.

1. The system should have the provision to track whether a user \((A)\) came in contact, within a certain number of days in the past (as guided by the epidemiologists), either directly or through a chain of intermediate contacts, with another user \((B)\) who has been diagnosed positive. The risk-score (signifying the probability of being infected) should factor in various epidemiological parameters including (but not limited to), how directly (or indirectly) \(A\) came in contact with \(B\), the number of days that have elapsed since the first contact event (as well as each of the intermediate contact events for indirect contacts), the distances between every pair of individuals who directly came in contact with another person in
that chain of contacts, the duration of each of those contact events, etc. This requirement may seem mind-boggling, but it is in no way far-fetched. Imagine a situation, where $A$, who is infected but pre-symptomatic, happens to infect a co-traveler Asha, while commuting in the local train on her way to office. The same day, Asha transmits the virus to his office colleague Hari and from Hari after a few more hops and steps the contagion jumps and infects $B$ by that evening. In a few days, $B$ feels unwell, develops symptoms and tests SARS-CoV-2 positive, whereas it takes almost 21 days for $A$ to feel sick and develop symptoms. Eventually, she also tests positive. The rest of the people in that chain may remain asymptomatic or presymptomatic for some time and suppose most of them become infectious. Shouldn’t we, in this situation, expect that as soon as $B$ tests positive, the DCTS helps us to somehow trace back to all the members in that chain of contacts starting with $A$? None of the currently proposed automated contact tracing systems is equipped to achieve this functionality and in the worst case may not raise a flag for the next 21 days while $A$, Asha and many others continue to infect other people, unknowingly, when they come in their proximity.

2. There should be negligible false positive or false negative cases. We must point out that this condition does not imply that every contact marked “at-risk” by the system is expected to be diagnosed positive (or every person declared as “not-at-risk” by the system is expected to be diagnosed negative), but rather the system should behave as closely as possible to a human contact tracer in terms of arriving at a decision, given the same set of information about a person coming in proximity to an infected person.

3. The user should be allowed to choose any app approved by an administrative authority of the person’s country of residence based on his/her liking or usability preference and he/she should be free to choose a different app at any point of time. Irrespective of the app he/she chooses, a basic contact tracing service should be available to seamlessly handle the device-side or server-side contact tracing data in such a way that neither he/she should miss any critical alert (e.g., “at-risk” notification) nor the system should lose any data meant for others.

4. The system should be accessible from any type of mobile device (smartphone or feature-phone) irrespective of the model, manufacturer, or operating system, although some functionalities may be unavailable in some of those types, models, or operating systems. If a user has multiple devices, he/she should be able to designate any of those devices as the primary one for the purpose of DCTS. There must be a one-on-one correspondence between a user and his/her DCTS account.

5. There should be provision to use a basic set of functionalities of DCTS across regional, state, or national boundaries as long as the local authority participates in a federated system or at least allows access to a remote system owned by a different authority.

6. The framework should be robust against any single device or data repository failure.
7. The system should not seek any personal or personally identifiable data except any one personal data that would be securely used by the registration module to ensure that each user has a single account.

8. The personally identifiable data should be securely stored in such a way that no entity other than the registration module should be able to identify or use that; in case the registration module or database gets compromised, the system should have an in-built mechanism to either destroy the sensitive data or encrypt it in such a way that it becomes impossible to decrypt by any entity in the future.

9. The system should have the ability to mask the history of interactions in such a way that the creation of interaction graphs or social graphs becomes impossible from the available information.

10. The risk-score computation must be dynamic and reflect a recent evaluation (if not instantaneous) of the risk of infection for a user. Risk-scores should not be stored as a static value because a person who was considered at-risk yesterday may no longer be considered at-risk today.

11. The system should be intelligent enough to provide an individualized recommendation to a user based on his/her age, general health condition, travel history, etc., apart from the computed risk-score, about whether the user is at risk or not and if at risk what next step he/she should take. In this way, the system would behave like an expert human contact tracer and not share recommendations for next steps blindly. The system should also have mechanisms to learn and adapt as the system matures and as per the evolving understanding of the epidemiologists. Here, any additional data shared by the user must be handled in a secured manner without any breach of privacy.

12. Finally, the system should be useful to gauge the probability of a user of being at-risk-of-infection or at-risk-of-exposure in a way better than the random chance, even when very few users adopt or use the system.

Expected functionality from the perspective of health authorities could be as follows.

1. The system should be capable of quickly adapting to any change recommended by the health authorities with respect to the risk-scoring approach and algorithm. A more advanced system should allow the health authorities to carry out what-if analysis by changing one or more algorithmic parameters and thereby analyzing how that may affect the number of at-risk users starting from the overall system level down to small neighborhood level without revealing actual identities of any user.

2. Designated health agencies should have the sole authority to validate if a user has been diagnosed positive and hence whether he/she is allowed to notify the same to the system and carry out the next steps. The system should also have mechanisms using which the health authority can reverse the status to negative either retrospectively or from a current time onwards.

3. The health authority/designated health agencies should have authenticated communication channels through the system to communicate with the administrative authority as well as the epidemiologists.
4. Using the system, the health authority/designated health agencies should be able to disseminate additional guidance and advisories to the primary users to help them fight the pandemic situation in a better way.

Expected functionality from the perspective of administrative authorities could be as follows.

1. The administrative authority may logically be divided into two parts—one that has administrative jurisdiction with respect to the users from the political and legal perspective and the other with the administrative rights from the automated system perspective. Assuming that a country may allow multiple parallel DCTS to function, multiple system administration authorities would have to functionally roll up to the central administration which would oversee the legislative framework behind the system.

2. The system administrative authority should have access to manage various functionalities like
   a. Registration of the users
   b. Sharing necessary inputs and parameters to the device-side implementation (like the app) at the time of initialization
   c. Setting up the server-side implementation (for centralized or decentralized server-based systems) or distributed bulletin boards (for blockchain-based distributed and decentralized systems)
   d. Changing of configurable parameters including the risk-scoring algorithm based on recommendations from epidemiologists and/or health authority
   e. Gracefully shutting down the system at the end of the pandemic
   f. Purging of user-specific data at regular intervals based on epidemiological recommendations or upon explicit user request
   g. Roll out system upgrades
   h. Storage and/or dissemination of protected yet sensitive data (e.g., pseudonyms of users who have been diagnosed as positive, risk-scores of users in case of centralized risk evaluation process, etc.) in the system in a secured way
   i. Communicate with federated system administrative entities of other countries or regions to evaluate the risk-score of its own user in presence of users visiting from other countries or help the other federated system administration to evaluate the risk-score of that foreign user.
   j. If there happens to be a standards body in future—maintain authenticated communication channels with their server(s) for information exchange.

3. The system administrative authority of DCTS must have a mechanism to interface with the central administrative authority to receive common inputs from it (like referring to the legal obligations that a user must have based on the laws of the land that gets reflected in the Terms and Conditions during the registration process) and sharing relevant information back to the central authority for the purpose of reports, reviews, verification, and auditing.

Expected functionality from the perspective of epidemiological authorities could be as follows.
1. There can be multiple epidemiological authorities who can query the system administrative authority to get aggregate level information (e.g., the number of infected users in a particular region, the density of interactions on an instantaneous and average basis, the probable hotspot regions, etc.) to track the evolution of the pandemic and measure the impact of control measures like lockdown, social distancing, quarantine, isolation, etc.

2. There should be a designated epidemiological authority (which can also be reassigned from time to time) that guides the system administrator to modify the epidemiological parameters (like how many days in the past the contact should be traced back for possible infection) and control the process of identifying at-risk users (by modifying the risk-scoring algorithms, etc.).

For the third category of users (malicious users), the ideal system must be robust enough to handle all possible adversarial attacks and thereby render the system completely unusable from the perspective of such malicious users with ill intent.

### 4.3 Attack Scenarios

An ideal system should be able to withstand all the following types of known attacks and possibly many more types of attacks that have not yet been discussed among the cryptographic community. The types of attacks can be broadly classified into the following three categories.

1. **Integrity violating attacks:** These are the attacks that mainly violate the integrity of a user’s (or app’s) state by introducing false positive or false negative cases, where a user’s app erroneously identified the user to be *at-risk* or *not-at-risk* respectively. So far in the implementation or in the literature of automated contact tracing, higher order contact tracing (i.e., tracing close contacts of those who have come in close contact of the infected individuals) has not been considered. However, if there happens to be such a DCTS, false positive may also include cases where the app incorrectly declares the user to be at risk after coming in contact with a certain number of at-risk users (depending on the risk-scoring algorithm). In this subsection, we would outline some of the examples of false positive attack types like replay attacks, relay attacks, and Inverse-Sybil attacks, and a few examples of false negative attack types, e.g., server data breach attacks and backend impersonation attacks. Note that there could be additional types of integrity violating attacks like tampering the proximity data stored locally in devices, that have not yet been discussed in the literature because a concrete approach to accomplish any such attack has not been identified. Similarly, there could be other means of compromising a device through ransomware attack, masquerading attack, etc., that may end up showing false positives or false negatives. There is also a possibility of a device being compromised by falling in the hands of bad actors due to coercion threats or device misuse attack. We are not discussing any such type of attack here that may arise due to virus, malware,
bugs in the operating system, or incorrect user actions (like clicking on an unsafe link or installing a compromised app).

2. **Privacy violating attacks:** In this class of attacks, a user’s personal or personally identifiable data may get leaked or compromised. Usually, a user’s phone number, location, address, contact details, interaction graph, etc., fall under the category of personal or personally identifiable data. If an app collects any such data then it can be susceptible to privacy-violating attacks. Even if the app does not use any such data and work with pseudonyms or pseudorandom numbers, there could be ways to track a user by capturing additional metadata beyond the app (like video footage or snapshots) and correlating or linking such metadata with app’s data for deanonymizing the user’s identity. There could also be ways to mount side-channel attacks (e.g., observing when a large set of data sets get uploaded from a device to a server or identifying the time-gaps between packets transmitted between neighboring devices) that end up disclosing sensitive information about a user. A large class of offensives that violate an individual’s privacy can be grouped under deanonymization attacks. Server data breach attack, device misuse attack, or coercion threats can also end up disclosing personal or personally identifiable data of a user.

3. **Usability-impacting attack:** If the usability of an app can be hampered or blocked by any attack, then that would fall under the class of usability violating attacks. A typical example of such an attack is called Denial of Service (DoS). Usability may also get severely impacted due to masquerading attacks, ransomware attacks, and deliberate injection of device or communication failures which are applicable in general for any mobile device.

Now let us discuss several attacks briefly.

**Replay attack:** A replay attack is a “person-in-the-middle” type of attack in which the received packets from a user’s (say A’s) app is captured by another user Carol who saves it in her device (through a compromised app) and later on replays the packet at a different location to B, Ted and Harry. Later, if A is found to be infected and is diagnosed positive, then the system may erroneously identify B, Ted and Harry to be at risk, although these users have never come in-proximity to any infected user.

**Relay attack:** A relay attack is similar to a replay attack, but it is expected to be executed in real time. That means between two devices, one or more malicious users may collaborate through a set of devices to relay the packets back and forth between the two devices at either end of that relay-chain. The effect is equivalent to that of the replay attack by introducing false positive cases in the system, however, relay attacks are inherently harder to detect and thwart as compared to replay attacks.

**Inverse-Sybil attack:** In this attack, a malicious user manages to use a number of devices through a single user identity. In this way, it may distribute the devices across different locations and increase the probability of coming in close proximity of an
infected user. In fact, it may replicate the user-id of an actual infected user and create a large number of false positives in the system because the app running on any other device coming in proximity with the compromised devices using the same user-id as the infected user may erroneously detect contact events. This attack derives its name from the Sybil attack. Wikipedia says

In a Sybil attack, the attacker subverts the reputation system of a network service by creating a large number of pseudonymous identities and uses them to gain a disproportionately large influence. It is named after the subject of the book Sybil, a case study of a woman diagnosed with dissociative identity disorder.

The Sybil attack may not have as acute an impact on DCTS as the inverse-Sybil attack. However, a large-scale version of the Sybil attack may disturb the epidemiological analysis of infection in a locality or region.

**False reporting attack:** If a malicious user is able to fool the system by falsely reporting diagnosed positive status while not being infected (e.g., forging the Health Authority’ approval token or stealing such a token before it gets detected) or manipulating the risk computation algorithm or the risk status of an app in any other way, it would be categorized under False reporting attack. Most of the DCT Systems are expected to be robust against this type of attack.

**Deanonymization attack:** In this class of attack, a malicious group of users on their own or with the help of colluding authority may manage to deanonymize the identities of infected users. The approach can be based on gathering a set of information at multiple locations and then correlating those data points to deanonymize user identities or getting privileged access to private or secured datasets or utilizing some metadata information (like video footage of customers in a shopping line) beyond the DCT System. The single-entry attack is a special type of the deanonymization attack, in which a user keeps his/her device near the device of a user (without physically coming in proximity) in a vulnerable place like hospital and switching off the app otherwise. Later, if the malicious user’s app declares the user at risk, then it would be easy to conclude that the original user must have been infected as the malicious user’s device was never brought in close proximity to any other device intentionally. There could be various other ways to achieve deanonymization. For example, in BLE packets, the MAC address field may be used to detect whether the packets are coming from the same physical device or not. Even if the address happens to change after some time, unless that change is synchronized with the change of proximity identifier, there can always remain some chance of linkability of two packets and hence tracking users through such linking.

**Server data-breach attack:** A DCTS may have different types of servers (backend servers, distributed hash-tables, health authority server, bulletin board server, etc.). If any of these server’s data gets breached, sensitive personal or personally identifiable user data may get leaked and/or compromised. Server data-breach may also lead to violation of the system integrity by having false positives or false negatives. In the
worst case, such leakage may lead to state surveillance, blackmailing by malicious actors, or receiving threats by terrorist or militia groups.

**Backend impersonation attack:** In this class of attacks, the backend (which may include the registration/initialization server, contact detection server, Health Authority server, etc.) being impersonated by a fraudulent server. The effects of such an attack can be on all fronts, i.e., violation of system integrity, violation of privacy, or disruption of system usability.

**Denial of Service (DoS) attack:** A typical case of a DoS attack may occur at the BLE layer itself where a group of hostile devices may keep pumping in junk packets to a targeted device and thereby overwhelming the app running in that device with the processing of useless packets. Due to such processing overload, the actual contact tracing process may get hampered—hence effectively causing a denial of service. There are certain attacks applicable at the Bluetooth layer (like Bluejacking, Bluesnarfing, Bluebugging, etc.), which may end up behaving like a DoS attack. There is also a possibility of a DoS attack that affects the servers or impacts the communication channel between the device and the server(s).

**Device misuse attack:** If an infected user who has been diagnosed positive deliberately allows his/her device to be placed in a busy location (like a marketplace) where the app continues to share and collect proximity data packets with a large number of other devices, that device may end up generating a large number of false positives in the system which could lead to a panic situation in that locality. The infected user may also attach the device to a moving vehicle (say a vehicle that delivers essential items) or even a street dog and, in that way, the false positive cases may come up at various places and not just limited to a single locality.

**Coercion attack:** This is a general class of attacks where a person may come under some threat by a bad actor or a group of people with malicious intent who may gain access to some personal or personally identifiable data through the app or abuse the data contained in the device to cause false positives or false negatives or some other type of panic in the system. There is also a possibility of that malicious set of people blackmailing or maligning the reputation of a person using the access to such a system.

**Miscellaneous types of attacks:** There can be a large class of attacks that can be mounted by utilizing the possible weaknesses of the cryptographic algorithms or leakage of secret key(s) of such algorithms that are used at various stages of communication or storage of sensitive data. We do not talk about those separately in this section. We also do not explore the possibility of complex attacks that can be engineered by rebuilding the global interaction graphs or social graphs around a set of nodes defined by pseudonymous identifiers. However, an ideal system must be able to overcome any such attack that has already been discussed in the literature or may come up in due course of time.
4.4 A Realistic and Aspirational System

Why is the ideal system not a realistic one? There are quite a few reasons. First and foremost, at the lowermost level of communication, automated contact tracing systems are proposed on top of Bluetooth Low Energy (BLE) technology which was not originally designed for contact tracing purpose and hence the expectation that it could be used for that application with the desired level of accuracy and precision can be questionable. Hence, the requirement that

The system should be accessible from any type of mobile device (smartphone or feature-phone) irrespective of the model, manufacturer or operating system.

is not realistic at this point. Similarly, the need that

The system should be useful to gauge the probability of a user of being at-risk-of-infection or at-risk-of-exposure in a way better than the random chance, even when very few users adopt or use the system.

could be infeasible at this point as governments of almost all countries have highlighted that the success of automated contact tracing system is singularly dependent upon the factor of how many users eventually decide to adopt and actively use the system.

In this section, we shall describe a high-level outline of a system that is both realistic and aspirational. The high-level outline will be divided into three parts—(1) an outline of a meta-framework and the need for standardization, (2) a generalized system that adheres to the meta-framework and proposed standardization, and finally, (3) an analysis of the proposed system from the cryptographic and architectural viewpoints. The reason for calling this system aspirational will be evident when we present the analysis, however, while describing a specific system we would show how the presently implemented and/or proposed solutions significantly fall short of the expectations when we compare those with the ideal system described in the previous subsection.

4.4.1 An Outline of a Meta-Framework

A meta-framework is a framework of frameworks. Here, we first describe the need for standardization from which the building blocks of the meta-framework emerge, followed by an outline of the meta-framework along with the components of standardization. Our idea is to provide a motivation and a direction rather than describe it in minute details. Moreover, one can always come up with a different variant and possibly an improved version of such a meta-framework.
4.4.1.1 Need for Standardization

Inherent non-homogeneity of any process necessitates standardization. In the absence of that standardization, the experience of users of that process suffers one way or the other. In case of contact tracing process, the inherent non-homogeneity stems from the following realities.

Coexistence of manual and digital contact tracing: In any country that has already rolled out or in the process of rolling out some digital contact tracing system, it would invariably co-exist with manual contact tracing process, simply because there are some people (e.g., children or aged members) in a family who may not be using separate devices and hence they would need to be traced manually once some other family member gets infected. Also, almost every country has an existing manual contact tracing department, its associated personnel, protocols, and procedures as part of its central health authority who might have so far used manual contact tracing in case of controlling any infectious outbreak in the past. It would be highly unlikely that they would dismantle such an existing infrastructure once DCTS is launched. Hence, the manual and digital contact tracing would co-exist. In that case, the risk computation algorithm of DCTS should be able to factor in the additional information available from manual contact tracing process. For example, a user’s app may decide that the user is not at risk based on available exposure data—but when that is combined with a manual contact tracer’s information of the user’s exposure through another infected user’s report (during interview), the combined exposure may put him/her in at-risk status. Similarly, there needs to be an understanding of how to resolve legal quandary. For example, if a DCTS user (say B) is reached out by a human contact tracer requesting him to get tested and he tests positive, would he be allowed to exercise his choice of not sharing the exposure database with the DCTS system’s backend or the bulletin board (whichever may be applicable)?

Coexistence of multiple digital contact tracing systems: Surprisingly, every digital contact tracing system’s design documentation has so far assumed that all users in the world would have an app as per that design. Even in the specifications that have talked about federated server systems in the backend (like BlueTrace or ROBERT), the assumption is that various countries would follow the same protocol and APIs, it’s only in the configurable parameters and in the app’s user interface level one may come across country-specific variations. In reality, the situation is vastly different. In Europe, in spite of the formation of Pan-European Privacy-Preserving Contact Tracing body, several competing specifications and protocols have already emerged and with the advent of Apple/Google protocol, the complexity has increased further with many countries deciding in one direction and then revising their approaches. The same has recently happened in the case of the United Kingdom as well which has now announced that they would adopt the Google/Apple protocol with certain features of their existing NHS_COVID-19 app based on their experience from the initial test roll-out. In the United States, the overall variations would be even more with each state independently deciding on their adoption strategies. With the gradual
removal of lockdown measures in different parts of the world and with more and more movement of people across borders, it would not be hard to imagine that soon people from different countries or even states would come in proximity to each other while running different DCTS apps on their devices which are not designed to talk to each other. If this is allowed to continue in this manner, the users would suffer big time as the apps would fail to notify them of the possible exposures. The health authorities and epidemiologists would also be unable to gauge the real-time situation of the spread of the pandemic as the aggregate data would be incomplete. In fact, it is not clear at this point in time if a country decides to switch its adoption from one DCTS system/protocol to another system/protocol what would happen to the existing data stored in users’ devices or in central servers or bulletin boards (wherever they are applicable). Without standardization of how multiple DCT Systems may talk to each other the entire effort of leveraging technology may fall flat.

**Differences in legal requirements:** The law of the land plays an important role when it comes to an app recommending the next steps, once he/she is diagnosed positive or identified to be at-risk. If A is traveling to a new country or a region, her existing app may not be equipped to handle the new norms and regulations as per the local laws around COVID-19. In the United States, there are state-to-state variations as well. For example, in the CDC site, it is mentioned in a page [3] that “Some states require mandatory testing for specific circumstances. Local decisions depend on local guidance and circumstances.” Hence, there must be some mechanism in the backend among the different administrative authorities of all the DCT Systems to communicate the specific legal restrictions of each region or country and the app must be designed to reflect the local context of the user dynamically.

**Additional differences:** There could be additional reasons as mentioned below.

- For example, a user may have multiple DCTS apps installed in her device and, at one point of time, she may use any one of those apps. For example, if A happens to travel frequently between her country of residence (say France) and her country of work (say Germany), she may find it convenient to install both the countries’ apps and use the French and German apps interchangeably as per her location dynamically, shouldn’t the two apps’ exposure databases have some means to communicate with each other?
- Another reason could be that there would eventually be some messaging-based systems designed for people who happen to use feature phones. The coexistence of such devices with smartphones would require the exposure information (to a positively diagnosed user of that system) to be accessible by DCTS user’s app and vice versa.
- If surface transmission gets captured independently by a different type of protocol or system in the future, there must be a way for DCT Systems to utilize that information in the background.
- We need not assume that all DCT Systems would be based on BLE in future as well. If any other technology (either already present in devices like WiFi scanning or a
new technology invented later by some manufacturers) then interoperability among BLE-based DCTS and non-BLE-based DCTS would also necessitate a standardization through a meta-framework specification. It must also be noted that currently even in BLE, different protocols are using different approaches—for example, some protocols are using BLE in broadcast topology mode (like Apple/Google, DP3T, etc.), while some protocols and apps are using the connected topology mode (like BlueTrace, TraceTogether, COVIDSafe, etc.). Even in broadcast topology mode, some are using the standard 31-byte payload (like Google/Apple) while some other protocol (like ROBERT) is using the Scan Response mode as the payload there is larger than 31 bytes. During the implementation of TraceTogether in Singapore, extensive calibration has been done with respect to the signal strengths for different device manufacturers (else RSSI values may not get standardized). Such calibration and standardization would be almost imperative for every type of DCTS implementation and the country to country implementation experience may vary.

Based on all the above, we can arrive at the conclusion that it is necessary to create a meta-framework that would provide the basis for standardization of design, implementation and roll-out of DCTS systems across countries as well as their interworking with each other and with the human-led contact tracing process in the best interest of regular users, health authorities, epidemiologists, and the administrators. We have already started observing the repercussions of lack of standardization (e.g., lack of adoption of some apps due to OS-related restrictions, reversal of strategy of which protocol or system to go for, etc.) in different countries.

4.4.1.2 The Components of Standardization

This subsection is called an outline of a meta-framework because we do not intend to present a full-fledged specification here. To create such a standard, it is important to organize a forum that can interface with different design groups, private and public organizations, research institutes and bodies dedicated to automated contact tracing solutions, authorities, and administration so as to ensure that pertinent details are not missed and also to have the requisite buy-in from the various stakeholders involved. Hence, we focus on describing a high-level view of the components that should ideally be part of such a generic framework or standard.

Risk modeling: The risk modeling should primarily be guided by epidemiological considerations. The basic unit of this model can be a contact event (or exposure event). Some of the pertinent questions that the meta-framework would need to answer are listed below.

- What are the mandatory parameters that a digital contact tracing system must consider (like duration of the two parties coming in proximity to each other, the minimum average distance between them)—could be measured in terms of signal
4.4 A Realistic and Aspirational System

Strength, etc.) two determine that an exposure happened? Are there any other recommended or optional parameters based on which the risk modeling may depend?

- What are the risk levels to be considered (at-risk and not-at-risk or high-risk, medium-risk, low-risk, etc.) at the minimum?

- How does the risk computation depend upon the possible direct contact events (in the last 14 or 21 days or so) with those individuals who have been diagnosed positive or indirect contact events with those identified by the system as at-risk (secondary contacts) or even tertiary order contacts? How should the risk computation factor in information coming through human-led contact tracing process or any other means?

- Would there be an optional provision to include possible additional false positives in a probabilistic manner to counter certain types of attacks?

- Which of the above parameters should be exposed to administrators as configurable so that the risk algorithm can be tweaked post-roll-out?

- What type of details (as mentioned above) would be transparent to the regular user?

- What kind of aggregate information be mandatorily exposed to health authority or epidemiologists?

- How should the risk be computed—dynamically at the time of the corresponding query or at a regular interval? Would there be a scope of storing such scores—in that case at what frequency would that score get refreshed? What are the optional choices for storage of such sensitive information and what would be the security requirements?

**Threat modeling:** The threat modeling is a security consideration. The basic units of this model are the given trust level of entities (e.g., trusted or semi-trusted backend servers), types of malicious entities, and the mandatory attacks against which the system must be robust. Some of the issues that the meta-framework would need to consider are as follows.

- What are the types of attacks against which a digital contact tracing system must be robust? What are the types of attacks against which the system must be resilient (e.g., even if there is a compromise of server data, the system can detect and overcome such issues in a certain time-frame)? It is extremely important for a system to adhere to certain minimum standards, otherwise, we may unknowingly accept a system vulnerability. For example, the latest BLE specification [2] of Apple/Google protocol has the vulnerability of deanonymization of infected users in the same way as depicted by Vaudenay in his analysis [7] of DP3T. In the absence of any standards, we have no choice at this point other than accept the decision of the two technical giants when it comes to using the Apple/Google protocol for Exposure Notification.

- If the system detects any threat attempted against it, then what kind of log should the system maintain in order to analyze, audit, or report such events to the system administration?

- How should the system handle the knowledge of new types of threats in the future? This question could be related to the version upgrade part of the standards.
Privacy modeling: This is also a part of the security consideration. In privacy modeling, the meta-framework should answer the following types of questions—

- What type of data must be considered as personal or personally identifiable by the system?
- What type of data must not be collected by the system even in encrypted or deanonymized form? What type of data must not be collected unless encrypted or anonymized?
- What are the mandatory constraints around the storing of personal or personally identifiable data that the system must follow?
- What kind of data of a user that must not be available to another user (even in encrypted form)? What kind of data of a user that must not be made available to another user unless encrypted or anonymized?

Registration step: This is an architectural consideration. This part of the standard should indicate mandatory constraints on (a) how many accounts (or user ids or login ids) an individual is allowed to create, (b) whether the person can access his/her account(s) from multiple (or different) devices, (c) what kind of personally identifiable information (like phone number or e-mail id) the system is allowed to cross-reference at the registration step for validation, (d) whether the personally identifiable information used during the validation step can be kept (in encrypted or unencrypted fashion) for the entire life cycle of the user in the system or if it is required to be deleted, (e) if a user loses the login credentials how can he/she retrieve the same, (f) how can a user access multiple login instances (g) what kind of mandatory parameters associated with the system (e.g., number of days the exposure data can be kept in the local device storage) should be shared with the app at the time of registration (h) any mandatory cryptologic requirement etc.

Proximity data sharing: This is an architectural consideration. This part of the standard should indicate mandatory constraints on (a) packet length, (b) coding and cryptologic requirements, (c) metadata, (d) unique service identifier to differentiate one system packet with another, (f) maximum data footprint size for an event, etc.

Contact detection and exposure data sharing: These are also architectural considerations. This part of the standard should indicate mandatory constraints on (a) max time-limit within which the contact detection must happen after an infected user (who earlier came in proximity to the current user) is known to be positive, (b) validity or time-limit of authorization token given by health authority in case of data upload, (c) coding and cryptologic requirements, (d) maximum data footprint for the related events, etc.

De-registration: This is an architectural consideration. This part of the standard should indicate mandatory constraints on (a) data cleanup and data retention process (connected with legal requirements as well), (b) reuse of login id by any other user
in future, (c) inclusion of required information (about the data) for aggregate computation purposes (to be later on referred by epidemiologists), etc.

**Fault modeling:** This is also an architectural consideration which should outline the mandatory requirements for system resilience against different types of failures and data losses. Incidentally, none of the currently proposed protocols and systems has talked about this aspect.

**Additional areas:** There are various other areas around which standardization would be required like (a) communication among backend servers (like system administrative server and health authority server), (b) communication among federated servers of the same protocol or system or between two different systems, etc.

The components of standardization can also be viewed in terms of network layers, like the following

1. **Physical and link layer:** The physical and link layer communication meta-framework involves the descriptions and usage of different physical layer technologies (BLE, Bluetooth, WiFi, etc.), the constraints of their usage and the definition of the hardware abstraction layer to create a standardized approach so that the higher layers may send/receive the packets in a way that is independent of the actual technologies being used.

2. **Network layer:** This is the main API layer that is used by the app which eventually gets translated in the form of link layer APIs that abstract the hardware interface. Since the hardware abstraction layer is not expected to impose any particular type of encryption/decryption or encoding/decoding algorithm, that critical functionality must be handled by the network layer APIs. So in a way, for any DCTS, this layer forms the heart of the security and privacy architecture and hence the meta-framework needs to define certain standard approaches that must be adhered to by all the DCT Systems in this layer.

3. **App layer:**
   a. **User interface part:** User interface part describes the way a user can make use of the contact tracing system, what other functionalities would be available to the user apart from the contact tracing and how the configurable parameters would be exposed by the system that requires user intervention and/or approval.
   b. **Client-server communication part:** This part is usually not transparent to the user. But important characteristics of the system in each phase of registration, initialization, de-registration, status notification, exposure detection, etc., can be standardized through this layer.
   c. **Server-to-server communication part:** A large-scale protocol standardization is required for defining the process in which heterogeneous systems can inter-operate across geographic or regional boundaries.

Any standard should also indicate optional as well as recommended parts. There could also be areas included as “must-not,” e.g., the standard may prevent the apps
to include any other notifications that go beyond contact tracing requirements and may be viewed as either propaganda or commercial advertisements.

4.5 A High-Level Description of a Generalized System

In this section, we provide a high-level description of a generalized system that would meet some of the expectations of an ideal system, which are not yet available in any of the designs described in Chaps. 2 and 3 under Centralized and Decentralized varieties. To maintain the brevity of the book’s structure, we describe an outline of the system and not a detailed design specification. Also, we assume that eventually there would be a standard meta-framework available which all DCT systems in the world would adhere to and hence our proposed system would be able to exploit certain features of that meta-framework. In absence of any such standard (or meta-framework), certain aspects of this proposed system would not be realizable or practically implementable. Since in this section, our intent is to offer a more generalized approach as compared to any special purpose architecture, let us call the proposed system, a Generalized Digital Contact Tracing System (GDCTS) or System-G in short that utilizes a meta-framework referred to as Framework-M.

We start describing System-G from a network layer perspective.

**Physical and link layer communication:** Assuming that the Framework-M would take care of the link layer protocol, System-G should not be dependent on a specific hardware interface (like BLE). The Framework-M would provide the required abstraction to hide the physical implementation details and expose a standard set of link layer APIs to the higher level layer (network layer). The Framework-M would be expected to be used by every other DCTS in the world so that the benefit of hardware layer abstraction is available to other systems as well. System-G would additionally exploit the link layer uniformity by implementing the system in such a way that the data packets from any other device in proximity, running some other automated contact tracing protocol that adheres to Framework-M, can be utilized and factored in to identify the risk of infection. In absence of a deployed Framework-M, the proposed System-G would be able to work only in a homogeneous environment (i.e., allow communication between devices that are running System-G-based apps only) and not in a heterogeneous environment where it may analyze packets coming from any DCTS app.

In the absence of any meta-framework and assuming that the physical implementation would utilize the BLE interface, System-G would have to define its own data link layer protocol. In case there exists a deployment of a meta-framework, the definition of the physical layer packets (using BLE technology) could also adhere to the following guideline.

- BLE interface would use the Broadcast topology and the packet size would be limited to the standard 31 bytes—hence there would not be a need to use the Scan-
4.5 A High-Level Description of a Generalized System

Response mode. The primary intent behind this guideline is to keep the device to device communication light-weight. Moreover, by avoiding the Connected topology or Scan-Response mode in the Broadcast topology, one may expect to achieve higher throughput.

- Each packet will contain 2–4 bytes of Encrypted Metadata (EM) apart from the pseudonymous pseudo-random ID (12–16 bytes) to store information like signal strength and a byte of information to indicate the maximum number of days the packet may be retained in the device. The service data part of the payload should also include a Message Authentication Code (MAC) so that data tampering can be detected. The idea is inspired in the way ROBERT protocol has incorporated MAC in its HELLO packet. However, we are not specifying the exact algorithm to be used for generating the MAC or proposing any specific length at this stage because the detailed design specification will eventually depend upon Framework-M as well.

- The advertisement and gathering of proximity data packets will happen transparently in the background. Currently, in Apple devices, no app is allowed access to BLE in the background mode. Therefore, it would be desirable that Apple and Google either come up with a meta-framework or align with the efforts that may be undertaken by a standards body to define a Framework-M. In absence of such an implementation from Apple/Google, System-G would have to abide by the current OS-specific constraints, which means that the app may not properly function unless it runs in the foreground. It must be pointed out that in almost all the mobile devices there is a provision to separately keep Bluetooth scanning mode ON even when Bluetooth is switched OFF. This is the configuration in which apps can continue to run in the background provided the OS layer supports the access to the BLE.

- The packet should have two separate indicators (a) a UUID value to signify the class of service (e.g., 0xFD6F can be used in line with Apple/Google’s for Exposure Notification Service ID) and (b) a sub-service SSID (included in the service data part), to indicate the exact protocol (this can be defined by some standards).

- There should be a provision in the implementation to store the packets in such a way that any DCTS app installed in the device can access the packets. Essentially, this is the concept of creating an abstraction layer through a meta-framework so that the storage of packets gets independently handled by a common proximity data access service. Of course, the packets of one DCTS system would not be decipherable by a different app, but the interoperability architecture will still ensure that such data can be used to measure the risk-score of the user. The proximity data access service should take care of deleting the old data from the local storage, based on information contained in the packet or a default number of days (say 21 days). Incidentally, this data store will be different from the app-specific data storage where the proximity data and meta-data would be stored after necessary decoding but the database will be encrypted so that no other app can decipher these information. The common data store will serve more or less as the backup for the app-specific data—even in the situation where the app may crash and the data becomes corrupt, the common store can be referred to retrieve past data when the app gets re-installed.
**Network layer:** This layer would consist of the APIs that would be directly used by the app to accomplish various functions like registration, initialization, creation, and communication of proximity data after applying the encoding and encryption steps, receipt, and retrieval of the proximity data from other devices after decrypting, decoding and safely storing those in the app-specific encrypted data store, uploading of proximity data for infected user or at-risk user (after validation by the Health Authority), checking the bulletin board or distributed hash-table to identify the risk of coming in contact with an infected user or even an at-risk user, requesting the restoration of status from infected, or at-risk to not-at-risk status (upon validation by Health Authority) and deregistration of user. These functions in the context of System-S are described in detail later in the next few subsections.

**App layer:** As indicated in the description of meta-framework, the app layer specification should be divided into three parts-

1. User interface part: The UI of the System-S would provide both diagnostic guidance as well as proximity-based risk notification to the user. The diagnosis-based risk assessment can be done through a series of questions posed to the user (it is possible to envisage that there can be an on-device AI engine that would learn from the epidemiological server the current disease transmission parameters and refine the diagnostic algorithm accordingly). In terms of contact tracing functionality, the UI should have provision to explicitly check with the user about his/her preference on
   a. Whether higher order contact tracing would be enabled (this is explained in the functionality wise description of System-S below)
   b. Whether to upload the proximity data in case of infected or at-risk status
   c. The choice of deregistration and withdrawal of participation at any point of time
   d. The choice of retrieval of backed-up proximity data in the device at the time of reinstalling the app (for some reason)
   e. The choice of participating in the federated interoperable DCT Systems.

2. Client-server communication part: There would be three types of servers with which System-S may need to communicate, namely
   a. The central server that would be accessed by the app (client) only when it needs to resolve the risk status for the owner of the packets that are emitted by a different DCT System
   b. The distributed hash-table based or the bulletin board servers (possibly using blockchains) which help the apps to resolve the risk status by peer-to-peer communication via the intermediate server (this part is entirely meant for the packets belonging to System-S-based apps)
   c. The server belonging to the designated Health Authority which provides validation of status change (between not-at-risk, at-risk, and infected states).
3. Server-to-server communication part: There could be four types of server-to-server communication, as follows
   a. Federated server to server communication (from the Central server to other Central servers) to handle the movement of users across multiple states or countries (including diverse protocol-based systems)
   b. Central server to bulletin board (or distributed hash-table-based servers) to resolve incoming query from other DCT Systems for risk status of the owner of a packet
   c. Communication of epidemiological data points between the bulletin boards (or distributed hash-tables) to the epidemiological database server (can also be a part of the Health Authority server)
   d. Communication of configuration parameters between the central administrative server and other servers/bulletin board (or distributed hash-table).

We now describe System-S in terms of the functional modules.

**Registration:** The registration process would ensure that (a) there is a proof-of-work check so that automated bots cannot create accounts, (b) each mobile device can have only one System-S user at any point of time (all past data stored in the device will automatically be deemed to belong to this user unless he/she explicitly raises a purge request, which would need additional validation steps), (c) the phone number is initially shared with the registration/admin server for validation that the user possesses the SIM card, however once a permanent user id is created at the back end with a mapping between login id of the user and the permanent id, the phone number is no longer stored and the local common storage will be created for the first time with that validation, (d) a person can access his/her account only from one device; from any other device the only option would be to create a new login id, (e) no personally identifiable information would be stored except the PIN/ZIP or equivalent code of the user that would be used later for epidemiological purposes, (f) if a user loses the login credentials he would have to create a new login id—however, the past device data and the corresponding user ids would automatically get associated with the same user, (g) a user can simultaneously access the system through multiple login ids provided he/she uses a different mobile device for each login.

**Initialization:** The registration step is immediately followed by the initialization step where the app received the configurable parameters from the server, approved URLs of the Health Authority’s server, bulletin board (or distributed hash-table) addresses, epidemiological server, etc., and also the relevant keys that would be used to have secure and authenticated communication between the entities.

**De-registration:** At the time of deregistration the device-specific local data would be purged and the mapping of login id with the permanent user id at the backend server will be deleted (hence the same login id can be reused later). The proximity data records stored in the bulletin board (or in the distributed hash-table) would not be touched as it would be impossible for the server to identify which records belong
to this user. Those entries will automatically get purged after the epidemiologically significant time-window is crossed.

**Risk computation:** This would primarily depend upon the number of close contacts (or exposures) in the past ESDays (Epidemiologically Significant Days) with either infected or at-risk contacts.

- The duration of the two parties coming in proximity to each other should be 10 min or more and the average distance between them (measured through calibrated RSSI parameters) should be less than 2 m.
- The risk levels would be at-risk and not-at-risk
- The risk computation algorithm will run locally and it will depend upon the possible direct contact events (in the last ESDays) detected by the bulletin board service (or distributed hash-table service) with those individuals who have been diagnosed positive or identified by the system as at-risk (secondary contacts). The actual algorithm would be downloaded by the app from the admin server once a week.
- There would not be any false positive deliberately introduced through probabilistic algorithms.
- The details of the number of contacts and types of contacts would be transparently available to the user and he/she may have an option to explicitly consult a health service professional or human contact tracer who would provide an independent evaluation of the user’s risk-status. In this evaluation, the health professional can factor in additional information like age of the person, past travel history, general health condition, etc. In fact, there can also be a possibility to expose an expert system-based interface to user, where he/she may play around different parameters and find out the way risk computation varies without disclosing any sensitive personal data.
- The risk would always be computed dynamically and this information would not be stored.

**Proximity data sharing between two System-S apps:** In System-S, we propose a symmetric approach of proximity data sharing between two apps running System-S protocols. It is inspired by the DESIRE protocol where an app comes up with a secret pseudorandom number in every epoch and shares a mathematical transformation of that secret number in the form of a rolling proximity number while broadcasting to nearby devices. When another System-S app running in a nearby device receives that number, it applies another mathematical function based on its secret number for that epoch and saves the Transformed Proximity Number (TPN). On the other hand, when the first app receives the rolling proximity number of that epoch from the latter device, it applies the same mathematical function before saving it locally. The functions are designed in such a way that these two saved numbers can be easily matched by an independent server or a distributed server (or a distributed hash-table) even though the secret keys are not shared by any of the devices. Before saving a TPN in the app-specific database, the corresponding metadata is decrypted and from the calibrated signal strength information (which should be part of the metadata), the approximate distance between the two devices is calculated. If the distance is more
than 2 m, the packet is silently dropped without saving. If the distance is acceptable, the TPN information is stored along with the decrypted metadata. If the same rolling proximity number is received by the app within one epoch (of duration 10 min) for at least 5 min, a corresponding flag is enabled in the data store to indicate that this encounter is epidemiologically significant.

The reason for keeping these two time-windows different and one with a value of 10 min, while the other with a value of 5 min, is to take into consideration that the epoch boundaries of the two apps may not be identical. If we consider that the 10 min of association of two individuals with an average distance at most 2 m is epidemiologically significant, there is a possibility of rolling proximity number of one app changing once during the 10 min epoch of the other. However, even with that change, there must be one time-window that is of the duration of at least 5 min. There is also a downside to this decision, as there could be some cases falsely reported as positive where one user may actually not stay for 10 min and still the system would flag the encounter as significant if the duration of association is around 5 min of more.

Proximity data sharing between a System-S app and a different DCTS app: If a System-S app receives a packet from a different DCTS app, it will not be able to decode that packet and hence it will simply store it within the local app-specific storage with additional metadata like the time of reception of that packet. Assuming that the received packet follows the SUID convention, it would be possible to identify the corresponding country/protocol combination that is represented by that SUID. Please note that here we are assuming a convention that the country identification is subsumed in SUID instead of following a convention of ROBERT in which the HELLO packet contains an Encrypted Country Code (ECC) separately. If a packet is present in the app-specific data store for more than $ESDays$, it would get deleted (by following a process similar to the one followed in device-specific common data-store).

4.5.1 Higher Order Contact Tracing

This is the functionality that is not present in any of the implemented or proposed systems that we have talked about in the previous two chapters. To the best of our knowledge, we have not come across such a proposal in the literature, so far. Through this functionality, a user can identify whether he/she has come in contact with some other user who has been identified in the system to be at-risk (and has not been diagnosed positive yet) in the past $ESDays$. We have described one such use-case while describing an ideal system. The motivation comes from the key observation of the epidemiological community across the world that apparent asymptomatic or pre-symptomatic people can also be infectious and it is important to contain the spread of the disease early enough by tracking such cases through extensive testing in the population or by launching aggressive contact tracing. If we consider a mathematical graph-like structure where each app acts as a node/vertex and we draw an edge between two nodes/vertices whenever two devices corresponding to two
nodes/vertices come near each other for an epidemiologically significant duration. In that case, one can envisage the set of nodes that are reachable by a direct edge from one node as that node’s immediate neighborhood. In that case, the problem of identifying the apps that have come in proximity of at-risk user apps is the same as identifying the neighborhood or neighborhood from a user’s node who has been diagnosed as positive. Theoretically, this can be extended indefinitely within the same mathematical framework, but in reality, we may not need to check beyond the set of users who might have come in second-degree contact with an infected user. This is the functionality that we refer to in this section as higher order contact tracing.

**Infection notification:** If a user is diagnosed positive, he/she gets a token in the form of a code (like a QR code) from a Health Authority (this part of beyond System-S). The token permanently changes the state of the app to `InfectedUserAppStatus`. This status can be reversed only after another diagnosis where the user is declared uninfected by the Health Authority (HA) and subsequently the user receives another token from HA to restore the app’s status to `UninfectedUserAppStatus` which is the initial state of an app at the time of installation. Whenever the HA releases a valid token, it keeps track of the zip/postal code of the infected user—this data is used by the epidemiologists to understand region wise infection count and disease dynamics. As soon as an app is assigned the state `InfectedUserAppStatus`, the user is asked if he/she would like to help others by uploading the proximity data received from other devices in a secured manner to the bulletin board server (or the distributed hash-table, etc.) so that other apps can check whether they came in proximity of the current app’s device in the past `ESDays`. If the user gives a go ahead all the relevant data gets uploaded after getting a blind signature from the Health Authority server to the bulletin board server. In addition, the user may agree that the app would continue to upload the additional daily received data till the point it changes state to `UninfectedUserAppStatus`.

**At-risk notification:** Whenever the user’s app determines (with the help of the bulletin board server) that a user is at-risk, the user would face another choice of whether he/she would like to upload the received list of proximity data to the bulletin board server so that other users can determine the events where they came in contact with at-risk users in the past `ESDays`.

It must be mentioned here that either during infection notification or during at-risk notification, the proximity data packets received from a different DCTS app are not uploaded. The processing of such packets is solely handled in the exposure detection stage as mentioned below.

**Exposure detection (homogeneous case):** At the exposure detection stage (in a homogeneous `System-S` app environment), each app uploads its received packets (for the past `ESDays`) once every day to the bulletin board server’s exposure matching detection service which not only detects the number of contacts with infected individuals but also the number of exposures with at-risk users. The consolidated data counts are shared with the app where the risk-scoring algorithm determines the
user’s risk after factoring in additional data points that may be present only locally at the user end and not stored anywhere else due to privacy consideration. The actual matching process is similar to that of DESIRE protocol and hence we are not going into the details any further.

**Exposure detection (heterogeneous case):** This is possibly the most complex and advanced functionality of System-S. However, it cannot work on its own in a stand-alone fashion unless there is a federated heterogeneous server to server communication infrastructure available through *Framework-M*. In the absence of such a support meta-framework or standardized server-to-server communication among multiple DCT Systems, the packets received from other DCT Systems would have to be silently dropped and ignored. If there is an interoperable system infrastructure supported by *Framework-M*, the following can be one way to handle exposure detection in heterogeneous systems

- The app will use the bulletin board or distributed hash-table server as a post-box to keep the list of packets received within last ESDays grouped according to SUID or country/protocol system names. The backend admin server will not know the app’s identity who is keeping the records. It would contact the admin or backend server of the other DCT System and resolve the question of whether any of these packets came from an infected user or not (belonging to that DCT System) and if true whether the duration of contacts was epidemiologically significant.
- Please note that for such packets, second-degree exposure detection may not be feasible as the server of that DCT System may not be maintaining such a list either centrally or in a distributed database
- If the other system is a centralized one (like BlueTrace), it could be easier to provide the required data back to the System-S admin server.
- If the received packet came from a purely decentralized DCTS, where the proximity matching service is transparently handled without the direct involvement of the admin-server, a special proxy service may need to be created at the server end to enable this kind of interoperability between heterogeneous DCT Systems.

### 4.6 Analysis of the Generalized System

We now present a high-level analysis of the system. First, we analyze it from the security and privacy standpoint and then carry out an architectural analysis.

#### 4.6.1 Security Analysis

**Replay attack:** *System-G* can not be subjected to Replay attacks because it is designed as a system that validates a match by symmetric exchange of packets (like DESIRE protocol).
**Relay attack:** It should be possible to execute a Relay attack on the system as inherently no location-specific metadata has been included in the proximity data packets. However, the restriction that a contact event will be considered valid only when the packets are exchanged at least for 10 min continuously between the two targeted devices, makes the adversary’s job complex. This is apart from the difficulty imposed by the symmetric nature of the protocol where the intervening set of devices have to relay the packets on both the directions to ensure that the contact event gets registered on both the devices.

**Inverse-Sybil attack:** Since the app can not be accessed with the same user-id from multiple devices, it would not be possible to subject *System-G* to Inverse-Sybil attack.

**False reporting:** We have not explicitly mentioned how the Health Authority may approve and validate the *diagnosed-positive* cases. Whenever the specific implementation of this generalized system defines the way the token would get generated, susceptibility of the system can be analyzed. One important characteristic of the *System-G* is that it also takes care of a user’s state restoration as and when the user is declared diagnosed negative. Hence, the possibility of a user indefinitely remaining in the system with *infected* status is no longer feasible.

**Deanonymization attack:** The proximity identifiers of an infected person is not made public in this system and the matching happens transparently at the backend/bulletin-board/hash-table-based server. Hence, infected user re-identification is not possible in *System-G*. It can still be possible through other means (like the reaction of a user on coming to know of the diagnostic test result or someone overhearing a conversation of a user with a health service professional), but that does not provide the malicious user with any additional advantage as any captured proximity identifier can not be replayed later to any other user to cause confusion and panic. Single-user attack can be prevented by incorporating a risk algorithm that does not give any indication of risk to the user unless at least two direct or indirect contacts with infected status get detected.

**Server data-breach attack:** No personal or personally identifiable information is collected and stored in the backend. Even the bulletin board data can not be deciphered without the help of respective apps’ secret keys. Hence *System-G* is not susceptible to the regular type of server data-breach attacks. However, in a heterogeneous system, the server to server data communication, if leaked, may reveal some information about the infected users.

**Backend impersonation attack:** Since we consider all the servers to use well-known domain names with the highest possible grade of security and authentication through valid certificates, this type of attack is not applicable on *System-G*.
4.6 Analysis of the Generalized System

**Denial of Service (DoS) attack:** There is nothing in System-G that can inherently prevent a typical DoS attack. In a specific implementation, we recommend that some deep learning-based solution should be added that must be first trained to identify junk BLE packets and then detect their presence whenever such attacks are mounted against the system.

**Device misuse attack:** System-G is definitely susceptible to such an attack. To prevent such an attack, there must be a mechanism (beyond DCTS) or a process to enable location congruity tracking based on the user’s self-declaration.

**Coercion attack:** Like Device misuse attack, Coercion attack is also feasible. Hence, in that way, *false positives* may get created in the system easily. However, there is not much risk of losing personal or personally identifiable data as even the user of the app can never decipher any sensitive information on his/her own.

**Social graph:** The backend server/bulletin board server would not be able to create the social graph as the proximity identifiers uploaded by a user once diagnosed as positive vis-a-vis by the same app for checking its exposure are not the same and hence can not be used to construct a social graph.

### 4.6.2 Architecture Analysis

**Extent of decentralization:** The specification is given in such a way that the system can be customized to the extent required to either a fully decentralized system or somewhere in between centralized and decentralized systems.

**Fault tolerance:** The system can withstand single-point failure at the device end where a common data store is created that remains separate from the app-specific data store. The distributed hash-tables or bulletin board servers can also be created in the cloud to have in-built fault tolerance against any single database failure.

**Interoperability:** The usage of Framework-M in System-G mainly serves the purpose of interoperability in a heterogeneous environment with multiple countries or regions/states.

**High-order contact tracing:** This is a novel introduction which has not been proposed in any other system so far to the best of our knowledge.

**Risk-computation algorithm:** This has been designed in a flexible yet comprehensive way so that the final recommendation of *at-risk or not-at-risk* status can be as individualized as possible without sacrificing the privacy requirements of the system.
The expectations of an ideal system from health authority, epidemiological authority, and administrative authorities can not be studied or analyzed at this point for System-\textit{G} as specific implementation details would be required to guide the way these authorities may access the exposed APIs of a system.

### 4.7 Conclusion

Writing a book on digital contact tracing systems in the midst of a life-changing pandemic feels like designing a flight-simulator for engines that have just begun their maiden voyages in turbulent weather. We do not claim that we know the final answer on what would eventually work. Therefore, as authors, we view this book as an earnest attempt to take a holistic view of this important topic with a specialized lens of cryptology. It is altogether possible that the terrain of automated contact tracing solutions would look much different after a few months and, in such a case, the later editions of the book would incorporate such changes. However, if one has to take a look at a snapshot in the time of how a set of system requirements, definitions, and designs on this important class of problem are emerging, then this book can be a reasonably good start. Moreover, in this book, we have tried to look at the problem from a medium to a long-term time-frame and hence the big picture view has enabled us to generalize certain solutions and point out the possible directions of the journey by taking into consideration the interests of multiple stakeholders, including—administrators, health authorities, epidemiologists, designers, technologists, and finally and, most importantly, the end users.

### References

1. Andrew, Ng. Why AI is the new electricity? Retrieved March 11, 2017. https://www.gsb.stanford.edu/insights/andrew-ng-why-ai-new-electricity.
2. Apple and Google. Exposure notification – Bluetooth specification, v1.2, April 2020. https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-BluetoothSpecificationv1.2.pdf.
3. Centres for Disease Control and Prevention. Contact tracing for COVID-19. https://www.cdc.gov/coronavirus/2019-ncov/php/contact-tracing/contact-tracing-plan/contact-tracing.html.
4. Fan, S. Contact tracing is the next step in the COVID-19 Battle – But how will it work in western countries? Retrieved April 21, 2020, https://singularityhub.com/2020/04/21/why-bringing-contact-tracing-into-our-digital-world-is-hard-and-why-we-must-do-it-anyway/.
5. Kurzweil, R. The singularity is near. Penguin USA, 2006. https://www.amazon.in/Singularity-Near-Humans-Transcend-Biology/dp/0143037889.
6. Tognotti, E. (2013). Lessons from the history of quarantine, from plague to influenza A. \textit{Emerging Infectious Diseases, 19}(2), 254–259. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3559034/.
7. Vaudenay, S. Analysis of DP-3T between scylla and charybdis. Cryptology ePrint Archive, Report 2020/399. https://eprint.iacr.org/2020/399.pdf.