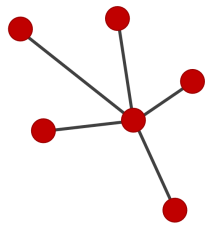


ABETS - A Better Exposure Tracing System

6.033 Design Project Preliminary Report

Enrique Casillas, Itamar Chinn, Daniel Stein

Massachusetts Institute of Technology

March 30, 2021

Contents

1. Introduction

During the COVID-19 pandemic, quickly and accurately tracing exposures to infection has become a critical task to limit disease spread within communities. Several existing systems already use smartphone technologies to automate this process, with each system contending with trade-offs between functionality, scalability, and privacy. Existing systems either collect a large amount of personal information at the expense of user privacy or prioritize privacy at the expense of functionality. Our goal is to design A Better Exposure Tracing System (ABETS) that meets the requirements of a university. ABETS integrates several different components — a central server and a set of Wifi routers operated by the university, as well as a large number of individual smartphones — to support a few key functions. First, the system *identifies contact events*, where individuals are close together for an extended period of time, using Bluetooth Low Energy (BLE) communications between phones. Second, it *notifies infected individuals* of positive test results via the central server, providing support as necessary. Third, using known contact events, it *determines individuals who were exposed* to an infected person and notifies them so they can begin isolation. Fourth, it *provides information for medical support and public health needs*.

ABETS supports these functions while prioritizing the design principles of *scalability*, *reliability*, and *user privacy*. Scalability means our system is able to handle a large number of users and situations in which a large number of positive cases arise simultaneously, whether these are spread throughout the community or concentrated within a cluster. In our design, demands on the server and on the network grow linearly in the number of users. Additionally, the demands on each individual smartphone are small and constant even as the total number of users increases. Reliability means that our system fulfills its functions in an accurate and timely manner. All positive cases in our system are handled appropriately, with the necessary notifications being made automatically within an hour. Finally, our system supports all of this while protecting user privacy, with only the minimum necessary information being collected and stored on the server. Importantly, all contact events are kept anonymous and maintained separately from the university's database of user information. In the following sections, we describe the implementations and

interactions of our modules, showing how they achieve these key design goals of *scalability*, *reliability*, and *user privacy*.

2. System Design

2.1. Overview

The diagram in Figure 1 outlines the main system design. It shows the main modules and components, along with the relationships between them.

The overall implementation takes advantage of three major components. At the lowest level, ABETS uses mobile smartphones to identify and communicate contact events and infection notifications using Bluetooth Low Energy (BLE). At the second level, the system uses routers to serve as packet switches in the communication network. Routers enable direct communication between the phones and the central server through Wifi and through ethernet, and offer a scalable, distributed approach to communicating data. Data from phones are also communicated anonymously using *onion routing* through multiple routers (see Section 2.4.2). These two methods constitute the communication module between phones and the central server. At the highest level, the system uses a central server that handles much of the storage and computation, including receiving data from university services and phones, processing and storing information, and sending data out to devices. The server houses an anonymized *contact graph database* used to determine exposure events as well as a separate *user information database*. The distinction between these two databases supports privacy within the system, as the server on its own has no way of determining who the individuals are within the *contact graph database*.

Contact events are obtained by phones through BLE signals while shared classes and living units are stored in the *user information database* on the central server; therefore, a key challenge was determining where and how to link these disparate interactions while still maintaining privacy. This had important implications for scalability and privacy. In our design, the *contact graph database* is stored on the central server for efficient computation, but all of the contact events are completely anonymized to maintain user privacy. Each phone generates its own *anonymous ID* that it uses in BLE broadcasts to establish contact events and in communications with the central server. These anonymous IDs are used to identify nodes in the contact graph database, but their mapping to user identities are known only to each individual phone and not by anyone else in the system.

2.2. Central Server

In our system, the central server fills two key roles: the *storage of user data* — including both *user information* and the *contact graph database* — and the identification and notification of infected or exposed individuals. Centralization of these tasks by the server allows for scalability as more users join the system, and reduces burdens on individual phones.

2.2.1. Data Structures

An important feature of our design that allows for privacy is that user information and contact events are maintained in separate databases on the central server.

User information is stored in a relational database that keeps track of the name, ID, phone number, living arrangements, and associated classes for every person at the university. Because all this information is already known to the university, it can be used to quickly identify exposures on the basis of shared classes or living spaces. Positive cases are already known to the university, so this involves minimal infringement of user privacy. Additionally, the relational database includes information about the isolation or quarantine status of each person.

Contact events are stored on the central server as a graph database for scalability. If exposure events were determined by individual phones using their own contact logs, then each phone would need to know about all positive cases in the system, placing a large burden on each device and on the network as the size of the system grows. Instead, maintaining the full community's contact event graph on the central server allows the system to efficiently query a list of positive individuals to find exposures. It is important to note, however, that contact events stored on the server are completely anonymized. Each node has only an anonymous ID and infection status that are not linked to user information in any way. Additionally, each edge has only an approximate timestamp associated with it to allow for queries of recent contact events and removal of old data. In this way, the university has no way of knowing the identities of individuals within the contact graph or who is interacting with whom.

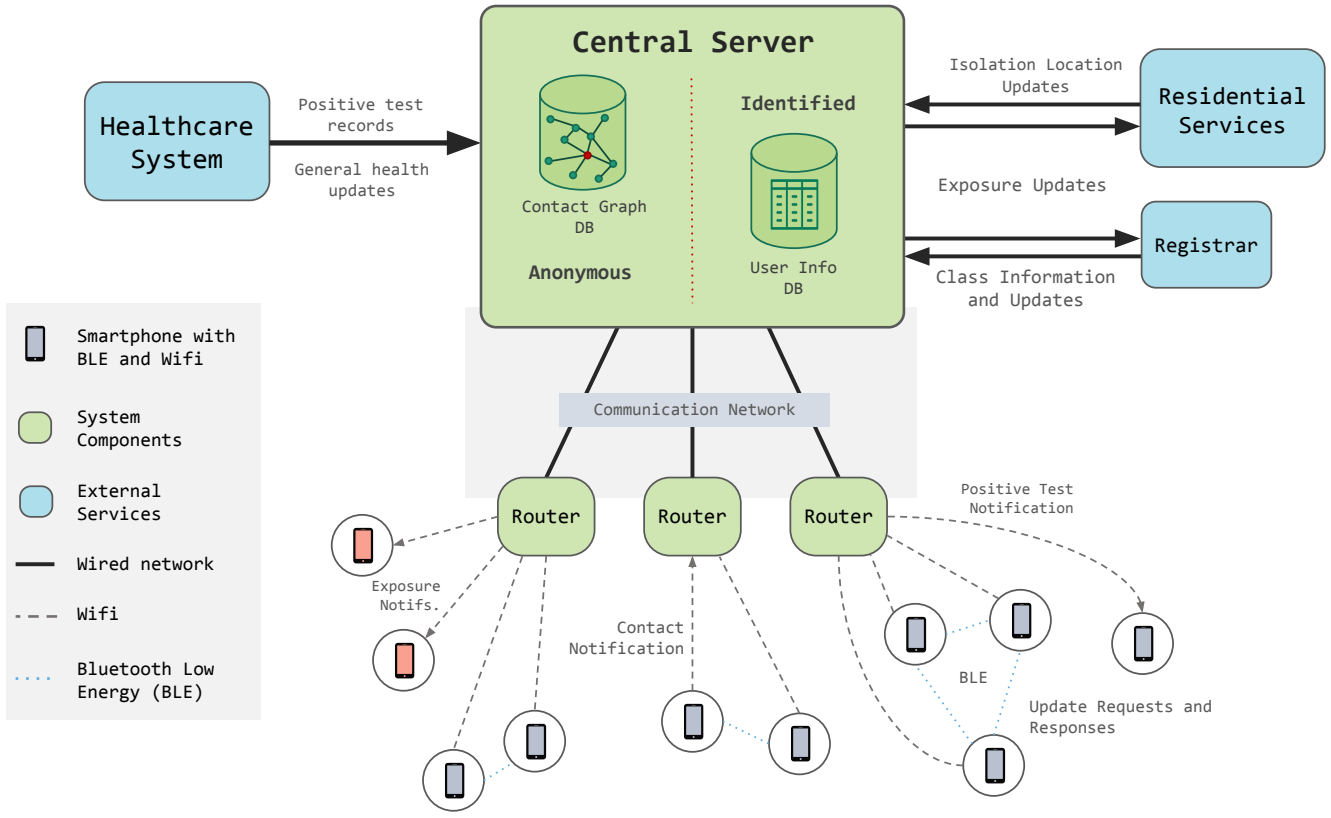


Figure 1: System Diagram

2.2.2. Updates to Server

Updates to the server can occur in three ways: 1) through updates from the healthcare system, 2) through updates from smartphones, or 3) through regular removal of expired data by the server itself. The healthcare system notifies the central server of positive tests to initiate exposure tracing. Smartphones communicate contact events to the central server, allowing the *contact graph database* to be updated. Additionally, the phone of an infected individual then notifies the central server of their anonymous ID and infection status to enable exposure tracing using the anonymized contact graph. Finally, edges are removed from the contact graph after 2 weeks, unless they involve an exposure event, in which case they are maintained for 180 days.

2.3. Phones

In designing the phone module, the central consideration was to preserve privacy, without compromising scalability. This section will discuss the design choices in the 'phone module', focusing on the phone-to-phone functions and data management.

2.3.1. Bluetooth Low Energy Communication

The phone recognizes *contact events* by listening for nearby phones' BLE broadcasts. All phones broadcast a 70-byte long message every 250 milliseconds, consisting of the phone's anonymous ID, a timestamp, and signal strength. All phones store the information in Table 1 in `list_of_broadcasts` for each BLE record they detect.

The function `determine_contact` recognizes if the same anonymous ID appears in `list_of_broadcasts` for 20 minutes or more at a high signal strength (under 3m) indicating a contact event. Eventually, both phones involved in a contact event must notify the central server for it to be added to the *contact graph database* (Section 3).

Field Name	Type	Description	Example
`anonymousID`	ID	64-bit ID number, representing the person of contact	da6af62d9-c574-4e9e-8 e62-cd549a722ee6
`contactTime`	datetime	The time when the contact occurred	"2021-03-19 11:53"
`contactStrength`	Real	A metric of distance between two phones, calculated by the phone itself	0.82133333

Table 1: Bluetooth Low Energy Log Data

The other function of the phone module is to periodically query the central server as to whether or not the phone's unique anonymous ID was a part of an *exposure event*, in which a neighboring node in the contact graph was infected. This communication happens using an anonymizing *onion routing* communication protocol (Section 2.4.2) which allows communication of the anonymous ID to the central server without exposing the identity of the phone user.

The full list of functions available to each phone module is shown in Table 2.

Function Name	Specification	Modules
`broadcast_ID (anonymousID, time)`	Loops every 250 ms and broadcasts a 70 byte message consisting of 'anonymousID' and 'time'	Phone to Phone
`receive_broadcast (anonymousID, message)`	Scans for nearby broadcasts and determines the signal strength	Phone to Phone
`determine_contact(list_of _broadcasts)`	Checks `list_of_broadcasts` for definite contact events	Phone
`send_contact_event (anonymousID, map_ anonymousID_time)`	Sends the central server a series of IDs with which the phone (and its anonymous ID) had a verified contact event with the time that event occurred (as defined by the first broadcast received). This occurs alongside the bihourly querying of the server for exposure events.	Phone to Server
`send_self_exposure (anonymousID, time)`	Sends the central server an update that `anonymousID` has been infected.	Phone to Server
`query_exposure_events (anonymousID)`	Queries the server to ask whether `anonymousID` has been exposed. This occurs bihourly.	Phone to Server

Table 2: Phone Module Functions

Two key design decisions were made in the phone module. First, we initiate most communications from the phone, rather than the central server. This allows us to prioritize privacy by separating anonymous IDs from users' identities through anonymous communication. Second, we store most of the data and do most computations on the server rather than on the phones. These decisions were made while evaluating three major use cases — few new infections; many new infections, updated continuously; or batches of new infections. The main weakness of our

design choice is that the phone will only be able to notify the user of potential exposure when it queries the server (bihourly). This is an acceptable compromise for preserving our key design goals of *scalability*, *reliability*, and *user privacy*.

2.4. Communication Protocols

To achieve the proper balance between functionality and privacy, ABETS uses two different protocols for communication. The first is a basic routing protocol where the server knows the MAC address of the phone it needs to send information to and sends messages directly to its destination. The second is for when the smartphones initiate contact with the central server in an anonymous manner, such as after a contact event. In order to help maintain the anonymity of the individuals using the system, messages are sent using *onion routing* to hide the communication source.

Section 3 describes in detail the situations in which each of these two different protocols are used within the system.

2.4.1. Public Server-Phone Communication

As is discussed in prior sections, ABETS requires that *positive test notifications* reported from the medical services are sent to the appropriate people and *exposure notifications* are sent to those in shared classes or living spaces. To accomplish this, ABETS supports a "public" communication protocol in which the central server can directly communicate with certain phones based on a known ID. This ensures the system operates quickly and reliably.

The public protocol is rather simple. Since the central server maintains a mapping of known user IDs to their phone's MAC addresses, it just sends the aforementioned updates out to the routers with the phone's address as the target, and the routers forward the message to its destination. This is done under a standard TCP protocol.

2.4.2. Anonymous Communication via Onion Routing

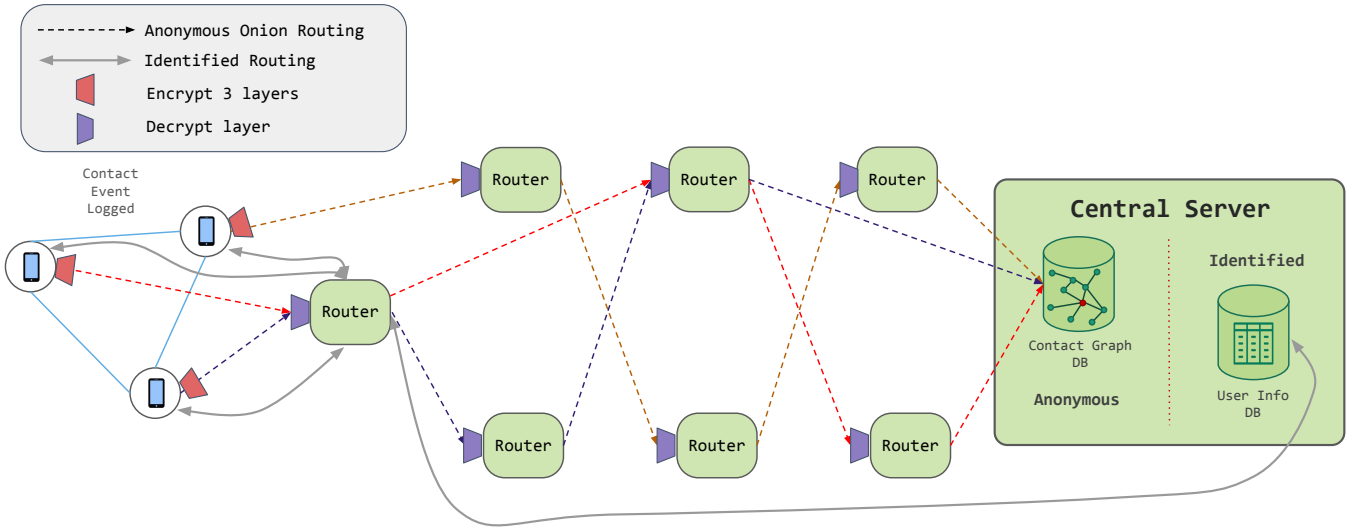


Figure 2: Onion Routing

ABETS requires that the update of an *contact event* be untraceable to a specific phone to maintain the anonymity of each phone's anonymous ID, and so we use *onion routing* to anonymize the source of the data communication (in the same manner as Tor [1]). *Onion routing* works by applying three layers of encryption, where each node decrypts one layer at a time, and identifies only the next node to forward the encrypted message to. The routers act as nodes in an overlay network. The identity of the updater is obscured by doing the initial three-layer encryption with three randomly selected nodes to forward the message. The last node has a fully decrypted message but does not know the origin of the message, and forwards it to the central server. This is illustrated in Figure 2.

When the central server module receives a new *contact event*, it responds by sending the phone module an

acknowledgement — in a similar process, the response follows the route back by hopping from node to node, where each node only knows the next node. The phone then deletes expired contact data from the phone, as this is now stored on the central server.

3. Use Cases

Now that the key protocols and modules have been established, we present the sequence of interactions involved in the four key functions of ABETS. The first function is identifying *contact events* between users and communicating those events with the server, which is the responsibility of the phone module. This can be seen in Figure 3a.

Notifying those *infected*, determining and notifying individuals who were *exposed* to an infected person, and providing information for *medical support and public health needs* are all largely the responsibility of the central server module. The general control flow of the system in these use cases is shown in Figure 3b.

3.1. Example Scenarios

Three scenarios demonstrate what could occur during the system's operational cycle, and how the system may respond.

The first scenario is *very low numbers* in which there is a low infection rate and there are few new daily cases. In this scenario, all 20,000 phones send periodic requests, but most are not infected. Of course, the system's goals would be fulfilled, though most phones would be sending what are essentially useless requests, wasting energy within the system.

The second scenario is *very high numbers* with a high infection rate. ABET operates well in this scenario, since positive tests are processed by the central server continuously. In addition, because of the randomized, periodic requests sent by phones, the expected number of packets passing through the network at any time remains relatively low.

Lastly, in a *compressed high numbers* scenario, positive test results are delivered in "batches" at the same time each day. This event presents the greatest challenge to the system, as it requires a large amount of processing power, and there could be high network activity in short periods of time as infected individuals are notified. Fortunately, because of the bihourly requests initiated by the phones (rather than the server), the strain on the network is mitigated.

3.2. Research and Public Health

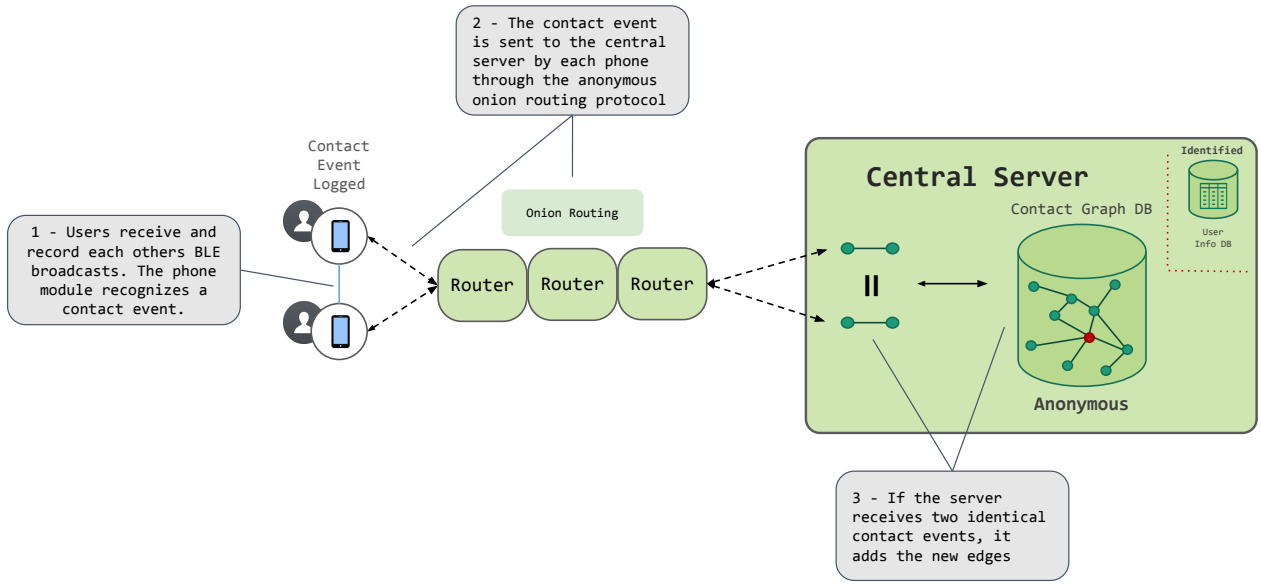
ABETS is also able to support public health and research needs, while maintaining user privacy. Because the contact graph database is completely anonymous it can be easily shared with researchers to study virus transmission within the community — such as the duration and closeness of contact necessary for transmission, or whether cases originate from different sources or from a single super-spreading event. We provide a function for the healthcare system and researchers to extract the contact graph over the prior 2 weeks, in addition to information about which nodes tested positive. The anonymous IDs used to identify each node are reassigned when the graph is exported to further anonymize user information.

4. Conclusion

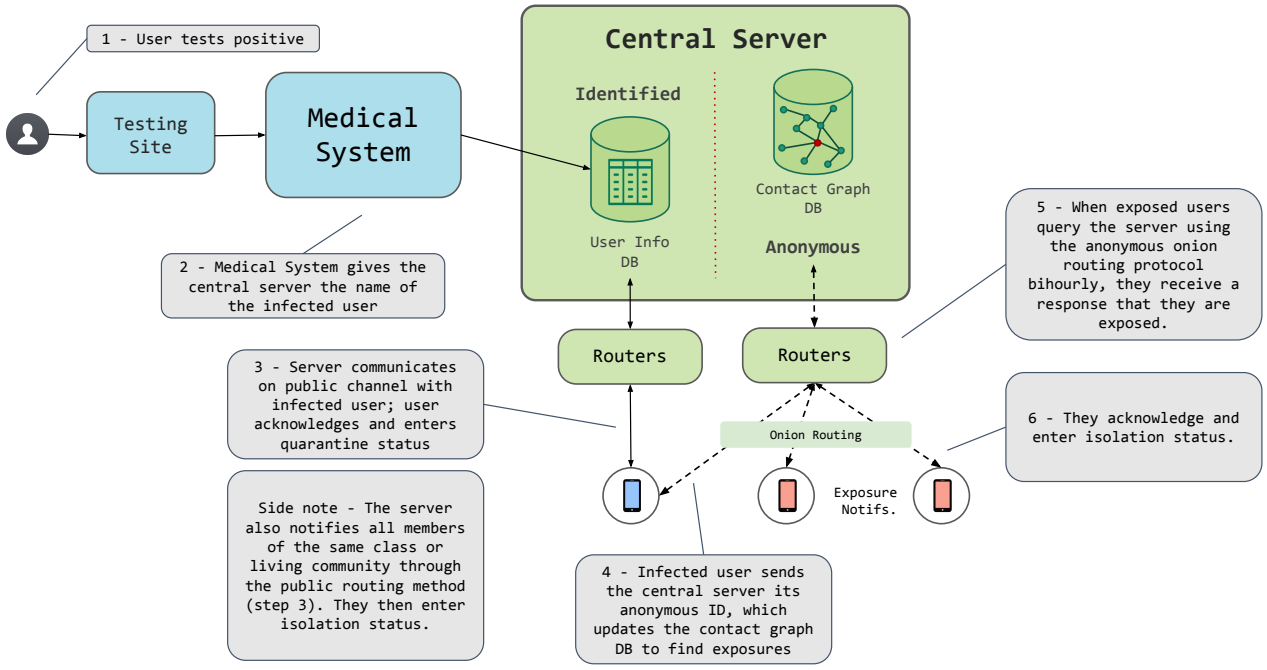
Our proposed system accomplishes the goals outlined in the specifications: it determines *contact events* and *exposure events*, notifying users of the need for isolation or quarantine; supports public health needs; and prioritizes user privacy. With most processing done on the central server, our system supports *scalability* as the number of users or infections increases. All functions are done automatically without need for user input, improving the *reliability* of the system. Finally, *privacy* is achieved through anonymity, both in the anonymous IDs used in the server's contact graphs and in the communication protocols used to relay information between phones and servers.

References

[1] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The second-generation onion router*. Tech. rep. Naval Research Lab Washington DC, 2004.



(a) Handling a Contact Event



(b) Handling a Positive Test and Exposure Event

Figure 3: Sequence of Events