TECHNICAL REPORT

ACOUSTIC-TURF: Acoustic-based Privacy-Preserving COVID-19 Contact Tracing

Yuxiang Luo, Cheng Zhang, Yunqi Zhang, Chaoshun Zuo,
Dong Xuan, Zhiqiang Lin, Adam C. Champion, and Ness Shroff
Department of Computer Science and Engineering
The Ohio State University

Abstract

In this paper, we propose a new privacy-preserving, automated contact tracing system, ACOUSTIC-TURF, to fight COVID-19 using acoustic signals sent from ubiquitous mobile devices. At a high level, ACOUSTIC-TURF adaptively broadcasts inaudible ultrasonic signals with randomly generated IDs in the vicinity. Simultaneously, the system receives other ultrasonic signals sent from nearby (e.g., 6 feet) users. In such a system, individual user IDs are not disclosed to others and the system can accurately detect "encounters" in physical proximity with 6-foot granularity. We have implemented a prototype of ACOUSTIC-TURF on Android and evaluated its performance in terms of acoustic-signal-based encounter detection accuracy and power consumption at different ranges and under various occlusion scenarios. Experimental results show that ACOUSTIC-TURF can detect multiple contacts within a 6-foot range for mobile phones placed in pockets and outside pockets. Furthermore, our acoustic-signal-based system achieves greater precision than wireless-signal-based approaches when contact tracing is performed through walls. ACOUSTIC-TURF correctly determines that people on opposite sides of a wall are *not* in contact with one another, whereas the Bluetooth-based approaches detect nonexistent contacts among them.

1 Introduction

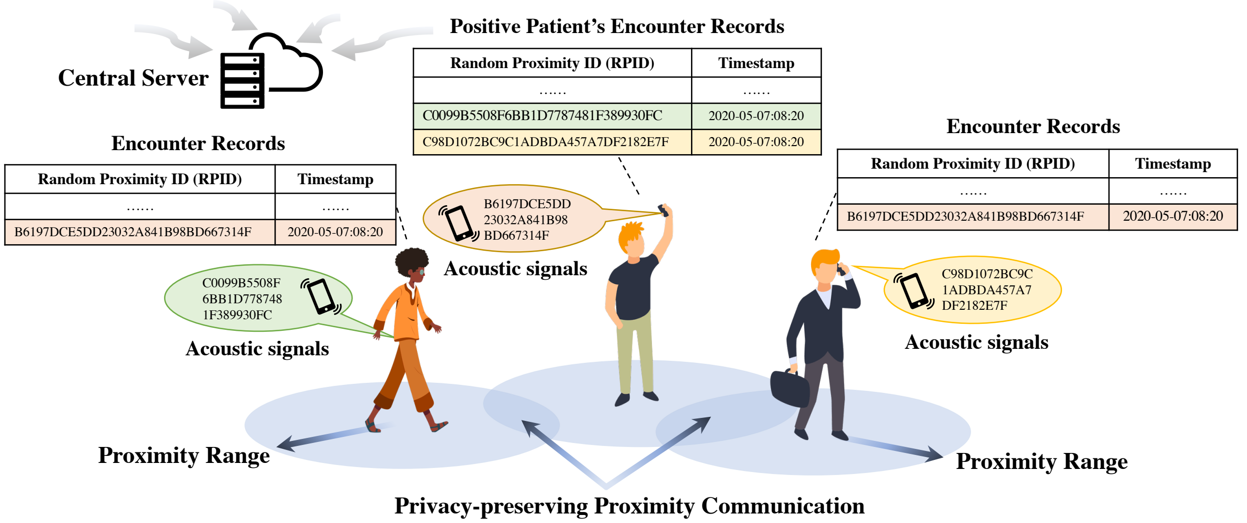
The outbreak of the novel coronavirus (COVID-19) has unfolded as an unprecedented worldwide crisis with enormous social and economic impacts. In many countries today, public health authorities have mandated *social distancing* (i.e., 6-foot separation, according to the CDC). Accordingly, travel has been restricted, schools have been shut down, shops have been closed, shelter-in-place orders have been issued, and remote work has become the norm. While vaccines or therapeutic drugs can fundamentally stop the COVID-19 pandemic, it is unlikely that they will reach the market soon (estimates range between 12 and 18 months away [3]). As such, we have to live with the virus in the interim, which necessitates techniques and tools for containment that also enable safe reopening of society and the economy.

At a high level, in addition to social distancing, practical approaches to contain the spread of the pandemic include (i) large-scale testing; and (ii) aggressive contact tracing, as countries such as South Korea [16] have demonstrated. Large-scale testing can quickly determine who has been infected and who has recovered, applying different strategies to respective groups; yet we must rely on medical science to solve the testing problem. Aggressive contact tracing can quickly identify those in the former group and recommend immediate quarantine within a certain period of time (e.g., two weeks) after a positive virus test result. However, manually performing such tracking would be impossible given large populations who may have been exposed to the virus. Hence, we need automated contact tracking systems.

In the past few months, researchers have made significant progress developing automated contact tracing systems using various location data generated from cameras, cellular towers, credit-card transactions, GPS signals, Wi-Fi hotspots, and Bluetooth signals. An effective automated contact tracing system must preserve the privacy of users; otherwise it risks becoming an invasive surveillance system. As Bluetooth achieves greater privacy and accuracy than alternatives such as GPS, it is widely considered that the use of Bluetooth signals is a practical approach [9]. Built atop Bluetooth, numerous privacy-preserving protocols including CTV [4], East-PACT [22], West-PACT [5], DP3T [25], COVID-Watch [26], and TCN [15] have been developed, and open source tools such as TraceTogether [18] and CovidSafe [7] have been released. To facilitate tracing app development and address other technical challenges (such as limited battery life), Apple and Google have jointly developed Privacy-Preserving Contact Tracing in the iOS and Android operating systems and provided corresponding API support [2].

While Bluetooth-based automatic tracking has shown great promise in fighting against COVID-19, it still faces two major challenges: large-scale user adoption and proximity accuracy. The efficacy of a tracking system depends on its use by $\sim$50–70% of the population [11, 26]. Designing a privacy-preserving, battery-saving system could help attract more users. However, achieving accurate proximity measurements via signal strength remains a challenging problem [8]. Specifically, Bluetooth's received signal strength indicator (RSSI) is used to measure the distance between two phones. In theory, stronger signals indicate closer phones. However, in reality, signal strength is affected by various factors such as phone orientation (i.e., portrait vs. landscape mode), surrounding objects and surfaces (e.g., indoors vs. outdoors, pockets vs. purses), and even phone operating systems (i.e., iPhone vs. Android) [8].

Therefore, integrating other mobile phone sensors is imperative in order to measure proximity precisely. In this paper, we describe how to leverage preexisting ultrasonic sensors in phones for automated contract tracing. Compared to Wi-Fi or Bluetooth signals, which are typically determined by hardware and influenced by the nearby environment, acoustic signals grant us a unique advantage: software can have a completely control over their communication. Based on this observation, we present ACOUSTIC-TURF, a new automated contact tracing system using acoustic signals from ubiquitous mobile devices without any additional hardware support. Similar to many other privacy-preserving systems, ACOUSTIC-TURF provides a high degree of



Random Proximity ID (RPID)	Timestamp
......	......
B6197DCE5DD23032A841B98BD667314F	2020-05-07:08:20

Random Proximity ID (RPID)	Timestamp
......	......
C0099B5508F6BB1D7787481F389930FC	2020-05-07:08:20
C98D1072BC9C1ADBDA457A7DF2182E7F	2020-05-07:08:20

Random Proximity ID (RPID)	Timestamp
......	......
B6197DCE5DD23032A841B98BD667314F	2020-05-07:08:20

Figure 1: Privacy preserving contact tracing of ACOUSTIC-TURF.

privacy in the following ways: (i) The communication (sensing) range of acoustic signals on mobile phones is naturally limited compared with those of Wi-Fi and Bluetooth signals; and (ii) each user's hardware IDs (such as Wi-Fi and Bluetooth MAC addresses) are neither used nor disclosed to nearby contacts. Instead, only randomly-generated IDs are disclosed to contacts, which guards against potential trajectory-based attacks.

An illustration of ACOUSTIC-TURF in action is provided in Figure 1. One salient feature of ACOUSTIC-TURF is that it detects contacts among mobile phone users in physical proximity. Specifically, assume there are users Alice and Bob. ACOUSTIC-TURF generates Alice's random ID and sends it to nearby users via acoustic signals. Bob, who is nearby, receives her random ID and stores it locally on his phone along with the timestamp; we term this a *contact record*. Later, Bob is diagnosed with COVID-19. He uploads his contact records of his choice to a central server maintained by public health authorities (such as the CDC). If Alice contracts the virus, the server can trace her contacts via Bob's uploaded contact records and notify her of the infection of a person with whom she had contact. In ACOUSTIC-TURF, these records do not reveal any Alice and Bob's identities; ACOUSTIC-TURF only reveals the random IDs of people with whom they were in contact at the corresponding times. Due to the range of acoustic communication, a listening adversary would have to be within this range to detect the random IDs, but ACOUSTIC-TURF updates them frequently.

In summary, this paper makes the following contributions: (1) To the best of our knowledge, ACOUSTIC-TURF is the first effort using acoustic-signal-based communication and sensing for COVID-19 contact tracing. (2) We design a novel acoustic broadcast protocol for working scenarios in which communications are casual, unintentional, non-cooperative, and opportunistic among up to ten people in occluded environments with limited power consumption. (3) We implement ACOUSTIC-TURF and its acoustic broadcast protocol on commercial off-the-shelf (COTS) mobile phones running Android. (4) Further, we evaluate ACOUSTIC-TURF's performance in terms of acoustic-signal-based encounter detection accuracy and power consumption under different ranges and occlusion scenarios. Our experimental results show that ACOUSTIC-TURF can successfully detect multiple contacts within a 6-foot range for both in-pocket and out-of-pocket scenarios.

The rest of the paper is organized as follows. In §2, we discuss related work including Bluetooth-based and ultrasonic-based proximity sensing in order to understand our techniques. In §3, we provide an overview

	Random Crypto Secrets Generation				Tracing		Reporting
Framework	At Client?	Random Secret (IDs)	# bits	Update Frequency	`setTxPower Level`	Proximity Measurement	Reported Item
CTV [4]	✔	Token	N/A	1 Minute	N/A	N/A	Token
DP3T [25]	✔	SK_t EphIDs	256 128	24 Hour 1 Minute	N/A	Threshold	SK_t, t
East-PACT [22]	✔	Seed Chirp	256 224	1 Hour n Minutes	N/A	N/A	Seed, t
West-PACT [5]	✔	S_0 S_i ID_i	128 128 128	Infection Period n Hour n Minute	MEDIUM	Threshold	S_i, t
TCN [15]	✔	RAK, RVK TCK TCN	256 256 128	Infection Period n Hour 15 Minute	MEDIUM	Customized Algorithm	RVK, TCK, t
Apple & Google [2]	✔	TEK RPIK RPI	128 128 128	24 Hour 24 Hour 15 Minute	N/A	N/A	TEK, t
ROBERT [23]	✗	ID EBID	40 64	Infection Period 15 Minutes	N/A	N/A	EBID, t
TraceTogether [18]	✗	TempID	672	15 Minutes	HIGH	Post-Processing	TempIDs
ACOUSTIC-TURF	✔	Seed RDID RPID	256 128 128	Infection Period 24 Hour 1 Minute	-	-	RDID, t

Table 1: Comparison of the recently proposed automated contact tracing frameworks. EphID: Ephemeral Identifier; SK: Secret Key; TCN: Temporary Contact Number; RAK: Report Authorization Key; TCK: Temporary Contact Key; RVK: Report Verification Key; TEK: Temporary Exposure Key; RPIK: Rolling Proximity Identifier Key; EBID: Ephemeral Bluetooth Identifier.

of ACOUSTIC-TURF, followed by detailed design in §4. §5 shares implementation details and §6 present the evaluation results. We discuss limitations and future work in §7, and we conclude in §8.

2 Related Work

Automated Privacy-Preserving Mobile Contact Tracing. Recently, numerous automated privacy-preserving contact tracing frameworks have been proposed. As summarized in Table 1, these frameworks use cryptographic techniques to generate various random identifiers (IDs), which they broadcast to nearby users via Bluetooth technology. One key approach to achieve privacy preservation is by periodically changing these random IDs such that an adversary cannot link them to specific users.

In particular, there are two types of approaches to generate random IDs: generating them on phones or on servers. Most frameworks generate random IDs on phones, except ROBERT [23] and TraceTogether [18], in which IDs are generated on servers. This leads to two different detection approaches: *decentralized* ones, in which each user's phone determines whether the user's proximity to an infected individual, and *centralized* ones, in which the central server determines user proximity. Most frameworks are decentralized.

In addition, frameworks use different names for random IDs and users generate these IDs using different cryptographic algorithms (e.g., hash chains in DP3T [25] and TCN [15], and HMACs in Apple and Google [2]). Random IDs are updated at different intervals. Except initial seeds (e.g., S_0 in West-PACT [5]) that never change during the entire infection period, the others either update when the Bluetooth MAC address

Approach	# Devices	Relative Motion Support	Distance (m)	Device Position
Hush [17]	2	✗	< 0.5	On obstacle
U-Wear [24]	> 2	✗	< 0.5	On body (fixed)
Dhwani [14]	2	✗	< 0.2[1]	No obstacles
DopEnc [28]	1–3	✗	1–1.5	N/A
ACOUSTIC-TURF	Up to 10	✔	2	Any position, occluded

Table 2: Comparison of multi-access communications using acoustic signals.

changes (e.g., RPI, TCN, and EBID), or are updated daily (e.g., TEK, RDID). In decentralized approaches, when infected users receive positive diagnoses, these users' random IDs are uploaded to the central server with corresponding timestamps, and these IDs are used to regenerate the broadcast IDs if the latter derive from the former. In contrast, centralized approaches upload contacted users' random IDs to the central server.

To determine if a nearby user is a close contact, all of the existing frameworks use Bluetooth technology, which many of our computing devices (including laptops, smartphones, and wearables) employ for short-range wireless communications. To measure proximity between two Bluetooth Low Energy (BLE) devices—a central one (such as a smartphone) and a peripheral one (such as another smartphone)—there are two fundamental sources of information:

- **Received signal strength indicator (RSSI)**. The most intuitive approach to approximate the distance from a receiver is to use the RSSI from the signal advertised from Bluetooth beacons [27]. Android employs the API `getRssi()` [21] by which a mobile application (app) obtains the received signal strength (in dBm), where the app measures distance based on a pre-computed threshold.

- **Transmit power (TxPower)**. A Bluetooth peripheral can specify the TxPower field in the broadcast packet to inform the receiver (e.g., the smartphone) of the BLE signal's transmit power. Android provides the API `getTxPowerLevel()` [20] for this purpose. Meanwhile, peripheral devices can use the API `setTxPowerLevel()` [19] to control the transmission power level for advertising that currently supports four levels: `ULTRA_LOW`, `LOW`, `MEDIUM`, and `HIGH`.

Therefore, when using BLE for contact tracing with smartphones, the sender smartphone can control the advertisement signal's TxPower and the receiver smartphone uses RSSI (and, optionally, TxPower) to determine a contact's proximity. Although employing both RSSI and TxPower seems appealing, this is actually a complicated process [1]. Through manual inspection of the source code of the available contact tracing apps, we have found that none of them use such an approach. In particular, as Table 1 shows, all available source code sets the `setTxPowerLevel()` parameter to either `MEDIUM` or `HIGH` transmission power. To detect whether a contact is in proximity, two apps use an `RSSI_THRESHOLD` value, and two others use custom algorithms.

Note that TraceTogether [18] does not directly determine whether there a contact is nearby. Instead, it uploads all metadata (including phone model and RSSI) to the back-end server, which determines proximity. Apparently, TraceTogether's developers made this decision since different phone models can yield varying RSSIs at the same distance with the same TxPower (as phones may employ different chipsets and/or antennas), according to data released on GitHub [6]. Consequently, TraceTogether performs calibration at the back-end, which indicates the complexity of the process of using Bluetooth for accurate proximity measurement.

Proximity sensing using acoustic signals. Acoustic signals, particularly using sound frequencies higher than the audible limit of human hearing (around 20 kHz), can also be used for proximity sensing and, hence, contact tracing. The basic principle is quite simple: the speaker in the phone sends out ultrasonic pulses and the microphone receives corresponding pulses. The ultrasonic pulses can be used to broadcast random IDs generated by cryptographic techniques, just as Bluetooth-based contact tracing frameworks use Bluetooth signals for the same purpose. Actually, modern mobile phones have all necessary sensors (speakers and microphones) available for acoustic-signal-based contact tracing.

There have been efforts to enable acoustic communications between devices that are equipped with microphones and speakers, as summarized in Table 2. In particular, Novak et al. presented Hush [17], a software modem that utilizes very high frequency sound to send data between commodity smart mobile devices. Hush performs fast modulation unobtrusively among users and achieves a low error rate. The system incorporates a fingerprinting scheme that hinders masquerades by attackers by allowing the receiver to learn and recognize packets sent from the intended sender. However, the communication distance spans only 5–20 cm, which is not feasible for accurate contact tracing. U-Wear [24] enables data dissemination between ultrasonic wearable devices. Dhwani [14] employs audible sound for near field communications between smartphones using OFDM and FSK modulations. This work focuses on high data rate and throughput for communications. DopEnc [28] can automatically identify persons with whom users interact and considers coordinating multiple access in order to measure the Doppler effect.

While there are efforts using acoustic signals for distance proximity by measuring the received pulses, they are inaccurate particularly in dynamic and noisy environments and cannot serve contact tracing well. In our Acoustic-Turf, there is no need to measure the received pulses. Compared to Bluetooth signals, programmers have far more control with acoustic signals. Note that in Table 2, we list Acoustic-Turf as supporting up to 10 devices tracing within a 6-foot range; this number is based solely on the practicality of how many people can be nearby within this range. The algorithm proposed in Acoustic-Turf is not limited to this number and it can support even more devices.

Most recently, concurrent to our work, Loh et al. [12] proposed integrating Bluetooth and ultrasonic signals for distance-aware automated contact tracing and implemented a system called NOVID. Since our Acoustic-Turf purely depends on acoustic signals for contact tracing, we wish to provide a detailed comparison with NOVID in this report. However, there is no public available technical details on how NOVID works. A detailed comparison will be provided once such details are available.

3 Overview

To address the limitation of low proximity accuracy when using Bluetooth technology, we propose Acoustic-Turf, an acoustic-signal-based approach for contact tracing. Acoustic-Turf has similar privacy-preserving aspects as many other frameworks (e.g., [4, 5, 15, 22, 25, 26]), but differs in how we determine proximity among contacts. Figure 2 presents an overview of Acoustic-Turf. At a high level, Acoustic-Turf has four stages (and eight steps): (I) initialization; (II) tracing; (III) checking; and (IV) reporting. We detail each stage (and correponding steps) below.

(I) Initialization. When a mobile user first participates in Acoustic-Turf, she downloads our mobile tracking application (app) from the corresponding app store (e.g., Google Play or the Apple App Store) and installs it on her phone. The installation (Step ❶) generates a per-device, unique, and random *seed* (s), from which we cryptographically derive the *random daily ID* ($RDID$). We derive the *random*

[1] Dhwani's short distance is for security reasons.

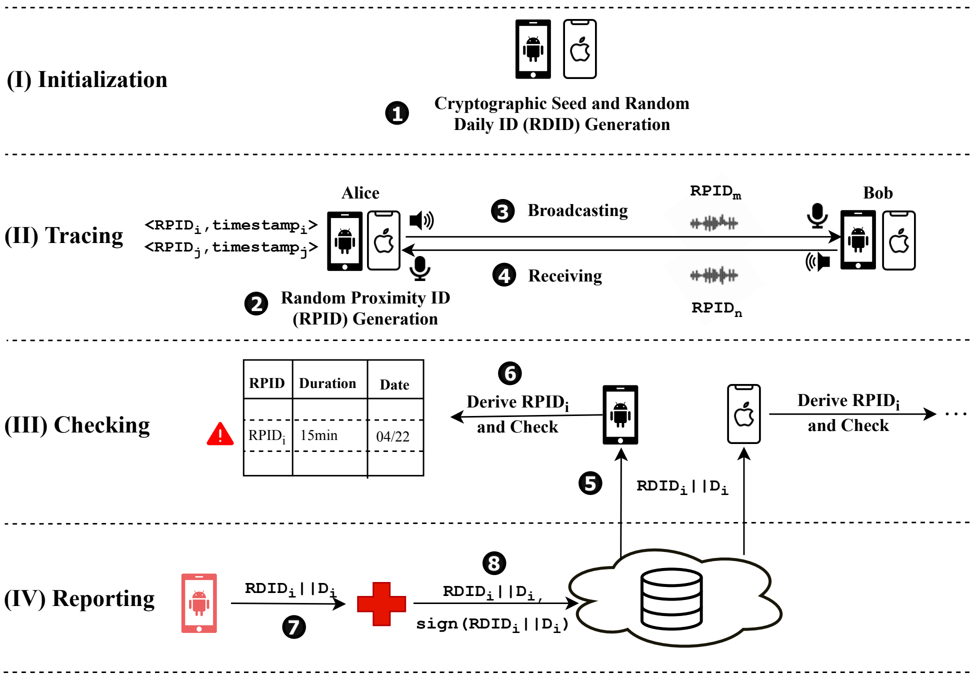


Figure 2: Overview of ACOUSTIC-TURF.

proximity ID ($RPID$) from the $RDID$. The system reports the $RDID$s to a trusted authority if the phone's owner is diagnosed with COVID-19.

(II) Tracing. When the tracing app runs on a particular day (e.g., D_i), it picks up the corresponding $RDID_j$, from which we generate $RPID$s (Step ❷). We choose to generate a random $RPID$ every minute in order to prevent nearby attackers tracking users. Next, a particular $RPID$ (e.g., $RPID_m$, 16 bytes of data) is broadcast via the phone speaker at certain frequencies to nearby phones during that particular time window (Step ❸). Meanwhile, the app receives an $RPID_n$ via the microphone from nearby phones and stores the $RPID_n$ in a log on the device along with the timestamp (Step ❹).

(III) Checking. While the app performs contact tracing, another thread periodically fetches positive $DRID$s along with corresponding dates D from the trusted authority (Step ❺). From these $DRID$s, the app derives all $RPID$s associated with that particular day D using the same algorithm as in the Tracing stage, and compares these $RPID$s with the stored logs on the phone (Step ❻). Crucially, we need to check the date, as an attacker can generate spoofed $RPID$s after learning a $DRID$ and broadcast these $RPID$s to victims, which causes false positives. Via timestamps in the log, we can easily calculate the period of time the devices (and their owners) were in contact with each other.

(IV) Reporting. Whenever a user is diagnosed with COVID-19, all of the user's $DRID$s and Ds for the past 14 days are uploaded to a trusted authority's server (Step ❼). To ensure data authenticity, we employ a healthcare authority to verify this information and sign it (Step ❽), such that adversaries cannot introduce false positives (e.g., by uploading their own $RDID$s) to mislead other ACOUSTIC-TURF users.

4 Detailed Design

In this section, we present the detailed design of ACOUSTIC-TURF. We first present how to use cryptograhic function to generate the random IDs used in our system in §4.1, and then describe the acoustic random ID broadcasting casting protocol in §4.2.

4.1 Cryptographic Random ID Generation

Cryptographic Seed Generation. To make ACOUSTIC-TURF simpler, all of the random IDs are derived from one *seed* (s), which is a per-device, unique, random number generated at install time. In our design, s is 32 bytes long, and generated by cryptographic *pseudorandom number generator* available in mobile operating systems such as Android. It never leaves the phone, and can be stored in a secure area such as TrustZone if that is available. It should also not be read by any malware.

Daily Random Identifier Generation. With the random *seed* (s), we can generate a large number of daily random identifiers ($DRID$s). In our design, we use as a hash function:

$$DRID_i = SHA256(\text{"}DRID_{i-1}||s\text{"}),$$

where s is our initial random *seed*, and the $DRID_{-1}$ is padded with 32 bytes of `0`s. We can generate a set of $DRID$s over a period of time (e.g., 14 days), as, currently, it is considered sufficient to only track the encounters over this time, though it can be changed as needed. The $DRID$s (each of which is 32 bytes) are used by the application (app) each day to generate the random proximity identifiers ($RPID$s), which are discussed next. The app can only access some $DRID$s when they are uploaded to trusted authorities (if the user tests positive for COVID-19).

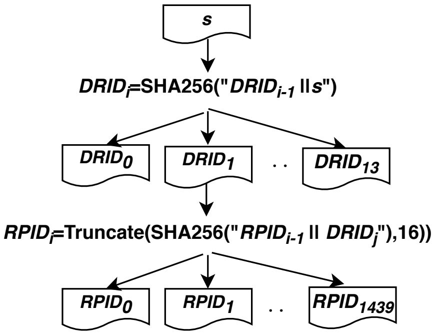

s
DRIDi=SHA256("DRIDi-1 ||s")
DRID0
DRID1
..
DRID13
RPIDi=Truncate(SHA256("RPIDi-1 || DRIDj"),16))
RPID0
RPID1
..
RPID1439

Figure 3: Cryptographic ID generation in ACOUSTIC-TURF.

Random Proximity Identifier Generation. When the tracing app runs on a particular day (e.g., D_j), it picks up a $DRID_j$ for that day and generates $24 * 60 = 1,440$ random proximity identifiers ($RPID$s) in Step ❷. To save bandwidth, we choose to use 16-byte RPIDs, each of which is also generated vai a hash function

$$RPID_i = Truncate(SHA256(\text{“}RPID_{i-1}||DRID_j\text{”}), 16)),$$

where $RPID_{-1}$ is padded with 16 bytes of `0`s. Each generated $RPID_i$ is broadcast at its particular time interval. Meanwhile, if a person is diagnosed with COVID-19, his/her $DRID$ is uploaded to the central server. All other participants in the system will retrieve them from the server, re-generate the $RPID$s based on the corresponding $DRID$s, and check whether they were in close contact to the infected user.

4.2 Acoustic Random ID Broadcasting

As discussed in the previous subsection, each generated $RPID$ is broadcast at its particular time interval. According to the CDC, users who remain within 6 feet of each other for more than 10 minutes face high risks of coronavirus infection [5, 13]. Hence, we must ensure that users receive at least one $RPID$ from one another if they have been in physical proximity for this amount of time, which is the basic requirement for our acoustic-signal-based random ID broadcast protocol. The time interval for $RPID$ broadcast is important too; clearly, it is under 10 minutes. With smaller time intervals, more $RPID$s are broadcast, which increases

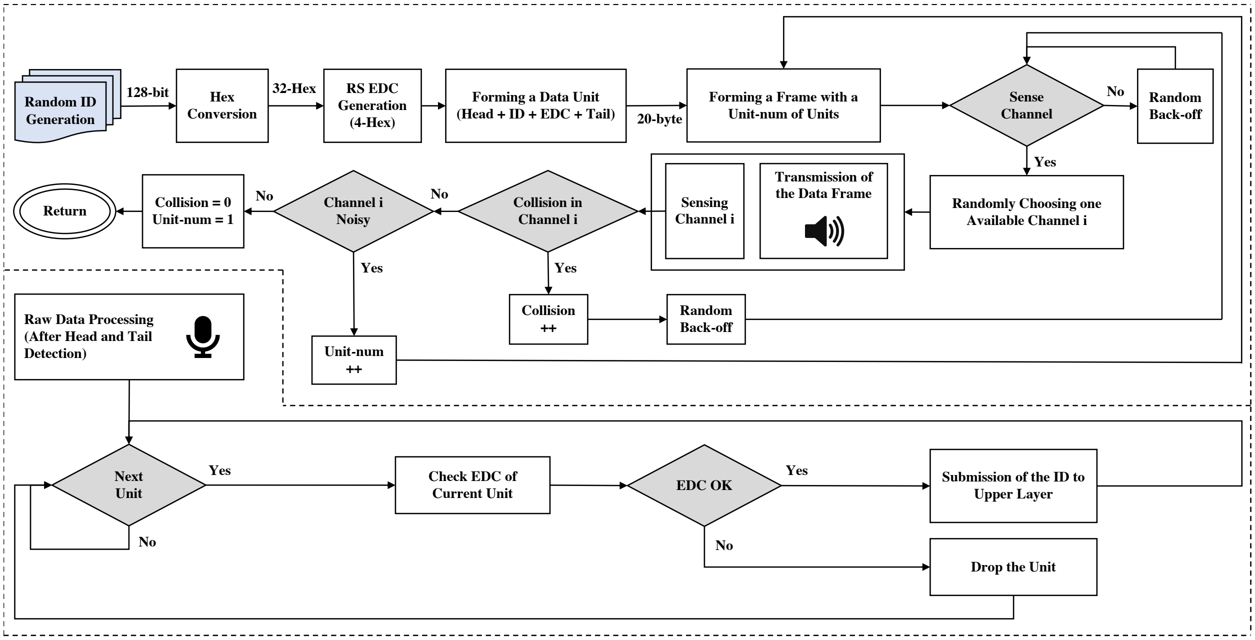

Random ID Generation
128-bit
Hex Conversion
32-Hex
RS EDC Generation (4-Hex)
Forming a Data Unit (Head + ID + EDC + Tail)
20-byte
Forming a Frame with a Unit-num of Units
Sense Channel
No
Random Back-off
Yes
Randomly Choosing one Available Channel i
Transmission of the Data Frame
Sensing Channel i
Collision in Channel i
No
Yes
Channel i Noisy
No
Yes
Collision = 0
Unit-num = 1
Return
Collision ++
Random Back-off
Unit-num ++
Raw Data Processing (After Head and Tail Detection)
Next Unit
Yes
No
Check EDC of Current Unit
EDC OK
Yes
No
Submission of the ID to Upper Layer
Drop the Unit

Figure 4: The random ID broadcasting protocol.

reliability at the cost of battery consumption. Users set the interval at their discretion; the default is one minute.

To meet the above basic requirements, our protocol must address several key challenges. It must realize acoustic communications among mobile phones up to 6 feet away in environments that may be occluded or noisy (e.g., phones in purses or pockets). This is hard due to mobile phones' limited speaker volumes and the nature of acoustic signal propagation. Also, our protocol must support various numbers of nearby users who need to exchange $RPID$s. Finally, we must conserve energy as phones have limited battery capacity.

Yet contact tracing, by its nature, affords us several opportunities. Communications among users are not interactive. The purpose of communications is to determine the existence of contacts and broadcast a user's existence by exchanging $RPID$s. Low data rates are acceptable and each user has a fixed-length (i.e., 16-byte) $RPID$ to broadcast. As we describe below, we leverage these opportunities in the design of our acoustic-signal-based random ID broadcast protocol.

4.2.1 The Broadcast Protocol

In the subsection, we describe our random ID broadcast protocol, which is an acoustic-channel multiple access control protocol. Our protocol adapts to changing channel and frequency occupancy, environmental noise, and distance among users by adjusting its communication channel, frequency, and reliability parameters. Figure 4 illustrates the protocol's workflow. In the following, we explain how the broadcast protocol works at the sender side and at the receiver side.

Sender Side. The data payload is a 128-bit $RPID$. First, the sender converts it to hexadecimal format using Reed-Solomon (RS) codes for error correction. Next, we encapsulate the $RPID$ with its corresponding RS code, a *head* symbol (`H`), and a *tail* symbol (`J`). [2] This is the basic data unit for transmission. The sender

[2] We describe the latter two symbols in the next section. We use these symbols to distinguish message boundaries.

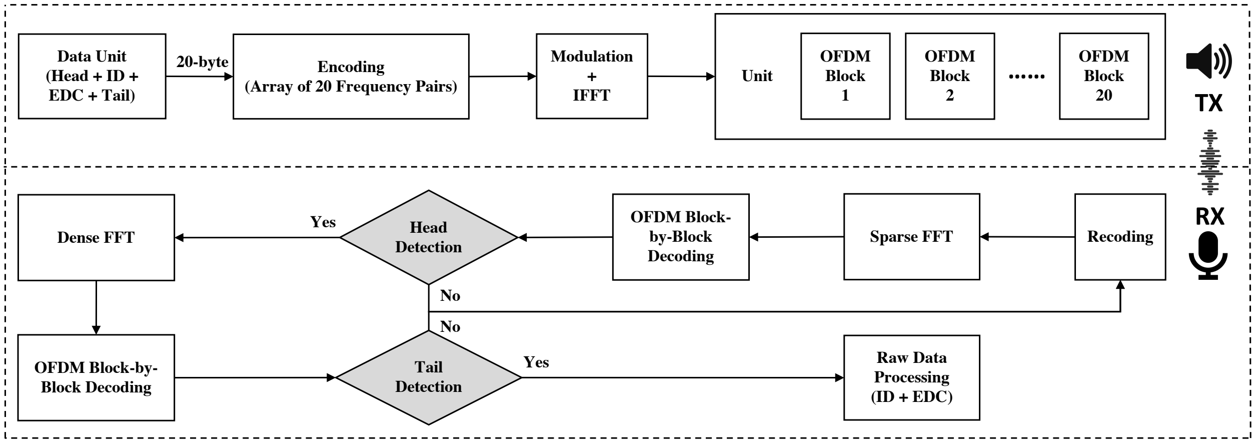

Data Unit (Head + ID + EDC + Tail)
20-byte
Encoding (Array of 20 Frequency Pairs)
Modulation + IFFT
Unit
OFDM Block 1
OFDM Block 2
......
OFDM Block 20
TX
RX
Recoding
Sparse FFT
OFDM Block-by-Block Decoding
Head Detection
Yes
Dense FFT
No
No
OFDM Block-by-Block Decoding
Tail Detection
Yes
Raw Data Processing (ID + EDC)

Figure 5: Acoustic data frame transmission and reception.

constructs a data frame by assembling one or multiple data units based on the number of units. After the data frame is ready, the sender scans channels for availability. If one or more channels are free, the sender chooses one at random and calls the data frame transmission module, which broadcasts the frame. Meanwhile, the sender senses collisions and noise on the occupied channel. If a collision occurs, the sender abandons the transmission, updates the number of collisions, backs off a random time based on this number, and scans again for available channels. If the channel is noisy, the sender abandons the transmission, updates the unit number, and forms a new data frame based on this number. Otherwise, the sender considers the data frame as successfully broadcasted.

Receiver Side. First, a receiver receives a data frame by calling the data frame reception module. The receiver checks each unit received from the module. If no errors are found, the receiver retrieves the $RPID$ from the data frame; otherwise the receiver checks the next unit in the frame.

Remarks. In our broadcast protocol, channels are dynamically selected based on occupancy. After the sender chooses a channel, the sender keeps sensing collisions and noise on the channel and takes action accordingly in order to avoid further collisions and achieve reliability without consuming too much power. Furthermore, the sender adjusts the number of units in a frame based on channel noise. The data unit is an important concept in the protocol (essentially, the unit contains an $RPID$ and an error detection code). At the receiver side, as long as one unit is correct, the receiver obtains the sender's $RPID$. Clearly, more units yield greater reliability. Adding units to frames does not require prior synchronization between the sender and the receiver, which is the benefit of our protocol design.

4.2.2 Acoustic Data Frame Transmission and Reception

In the above broadcast protocol, the sender calls the data frame transmission module to broadcast the data frame using acoustic signals, and the receiver calls the data frame reception module to receive a frame. We discuss these two modules, of which Figure 5 illustrates via flowchart. First, we describe our communications terminology.

Communications Terminology. The transmission module sends a frame with one or multiple data units on an acoustic channel. Each data unit comprises 20 bytes (i.e., 160 bits). We encode data using 4-bit symbols, where each symbol is a number, a head symbol (H), or a tail symbol (J). In our protocol, we send symbols over long periods of time for reliable long-range communication. We send data symbols in frames employing binary phase-shift keying (BPSK) modulation. However, sending only one symbol at a time via the carrier wave would limit network throughput. To increase it, we send symbols in *chunks*, packets of data containing two symbols sent in parallel. Our protocol supports multiple channels in total, where each user is allocated one channel at a time. To reduce interference, each channel is a combination of two different frequencies. Next, we discuss frame transmission and reception.

Data Frame Transmission. One data frame consists of one or multiple data units, each of which contains 160 bits (i.e., 20 bytes). In our coding mechanism, we organize each sequence of consecutive 4 bits as one symbol; hence, there are 16 different symbols. We modulate each symbol at a specific frequency using an Inverse Fast Fourier Transform (IFFT). Frequency selection follows that of OFDM. For example, when we send the symbol 0111 at frequency f, we modulate this symbol at the carrier wave frequency f. We place two symbols in a chunk; we modulate each symbol at a different frequency. The first and second symbols in a chunk correspond to an odd-numbered bit (modulated at a lower frequency) and an even-numbered bit (modulated at a higher frequency), respectively. Given a 160-bit data unit, we convert it to 40 4-bit symbols, which we place in 20 chunks. For each symbol, we select its frequency based on a hash table mapping symbols to frequencies. We delimit the sequence of symbols using the head symbol (H) and the tail symbol (J), respectively. We create a frequency-domain representation of the signal with 2^{15} (32,768) bits corresponding to the odd and even symbols. We perform an IFFT on this data to generate 2^{15} values using amplitude modulation (AM) in the time domain with a 48 kHz sampling rate. We truncate the generated audio signal using the symbol duration and emit this signal via the mobile phone's speaker.

Data Frame Reception. We receive an AM signal sampled at 48 kHz, which we process using a sparse Fast Fourier Transform (FFT) for rapid symbol detection. We perform another dense FFT upon detecting a head symbol. Specifically, we receive 256 samples in an array; there are many such arrays in sequence. Each time, we perform an FFT on 2,048 samples (a 42.6 ms audio signal). We employ a sliding window protocol with the symbol duration to scan the amplitudes based on the frequencies we selected for OFDM block-by-block decoding, If the amplitude at a given frequency exceeds a certain threshold, we regard it as a 1. To avoid interference, we scan the low-end and high-end frequencies separately. We start and stop collecting data upon receiving head and tail symbols, respectively. We place all received symbols in a frame, which we return to the mobile app at the receiver side.

Remarks. The above frame transmission and reception modules provide acoustic transmission service for a given data frame. To achieve reliable communication, we adopt several mechanisms, including long symbol duration, BPSK, OFDM frequencies, and parallel frequency use (at the cost of low data rate).

5 Implementation

We have developed a prototype of ACOUSTIC-TURF on Android. Our prototype consists of two major components: an ACOUSTIC-TURF mobile application (app) for each participant, and an ACOUSTIC-TURF server for public health authorities (e.g., the CDC). The server uses a database that securely stores all relevant information. We implement the acoustic random ID boradcast protocol at the mobile app side and the security protocol between the mobile app and the server.

The Mobile Apps. The ACOUSTIC-TURF mobile app is a downloadable app that periodically generates $RPID$s, broadcasts them, and receives them from other devices. The app consists of four fragments: Home, Your Contacts, Your Account, and Settings.[3] The Settings fragment lets users modify personal information and adjust app settings. The Your Contacts fragment contains $RPID$s received from nearby users during the last 14 days. The Home fragment has three pages: a Welcome Page, a User Portal, and a Notification Center. The Welcome Page briefly introduces users to ACOUSTIC-TURF and instructs them on how to use the app. Users can login or register though the User Portal, which synchronizes their account credentials with the ACOUSTIC-TURF database. In the Notification Center, users can view notifications and warnings sent from the server.

We generate the cryptographic *seed* via the Java class `java.util.Random`; we save *seed* via `SharedPreferences`, which constitutes private app-internal data storage that is off limits to other apps. Similarly, we store $DRID$s in private app-internal storage for each of the last 14 generations using the SHA-256 hash function. The *seed* is only needed to generate $DRID$s, which we use to generate $RPID$s periodically.

We implement our random ID broadcast protocol as well as the data frame transmission and reception modules. To reduce audibility, our implementation uses ultrasonic waves from 17.5 kHz to 22.6 kHz. In particular, it supports two acoustic channels, each of which uses 54 frequencies. The interval between adjacent frequencies is 46.875 Hz. The first two groups of 21 frequencies are assigned to deliver the odd and even symbols discussed in §4.2.2, respectively. We reserve the remaining frequencies to separate adjacent channels for reliable communications.

The Server. The ACOUSTIC-TURF server only stores $DRID$s and corresponding dates for users who have tested positive for COVID-19. Mobile app users periodically retrieve these $DRID$s to determine whether contact with these users occurred. For efficiency, the server only pushes newly added data to the client. Also, the integrity of the server data is protected, and we require authentication in order for users who are diagnosed positive to upload their $DRID$s to the server.

6 Evaluation

In this section, we present ACOUSTIC-TURF's experimental results. First, we describe our experimental setup in §6.1. We discuss the results of effective tracing distance in §6.2 and energy consumption in §6.3. Finally, we compare ACOUSTIC-TURF with Bluetooth solutions in §6.4.

6.1 Experimental Setup

We evaluate ACOUSTIC-TURF's ability to trace high-risk contact activities in two typical scenarios: *Noisy In-Pocket* scenarios and *Quiet Out-of-Pocket* scenarios, as environmental noise and occlusion are two major factors that affect ACOUSTIC-TURF's contact tracing quality. These two scenarios represent two extreme working scenarios of ACOUSTIC-TURF under the effects of noise and occlusion level. In other words, we believe that in the Quiet Out-of-Pocket scenario, ACOUSTIC-TURF receives the least interference, which would result in the longest tracing distance; on the other hand, in the Noisy In-Pocket scenario, ACOUSTIC-TURF experiences significant interference, which would result in its shortest tracing distance. We use the group tracing success rate (defined below) to evaluate the reliability of ACOUSTIC-TURF. Suppose there are n users in a group in physical proximity. For any user u_i, we denote the number of other users detected (received at least one $PRID$) by u_i in 10 minutes as k_i. We define the *individual tracing success rate* $R_i = \frac{k_i}{n-1}$, and the *group tracing success rate* $r = \overline{R_i}$.

[3] In Android development, a fragment is a composable piece of the UI that roughly corresponds to a single user-visible screen.

We also evaluate ACOUSTIC-TURF's energy consumption to determine how ACOUSTIC-TURF affects the battery life of mobile devices. We compare ACOUSTIC-TURF to Bluetooth Low Energy (BLE) in terms of its ability to permeate a wall in order to highlight ACOUSTIC-TURF's advantages in contact tracing against COVID-19. Throughout the evaluation, all mobile phones that run ACOUSTIC-TURF are Google Pixel 4s running Android 9+.

6.2 Evaluation of Effective Tracing Distance

To evaluate the effective tracing distance, we place four Pixel 4 mobile phones at the vertices of a square on a flat hardwood floor. All phones face up while their speakers all point north. Each phone keeps broadcasting one *RPID* at full volume for 10 minutes (the *RPID* changes every 10 minutes) while the data frame that carries one *RPID* is sent every 50 to 70 seconds. Each phone has a distinct *RPID*. We collect the number of *RPID*s received by individual phones every 10 minutes in order to calculate the group tracing success rate. We evaluate the group tracing success rate performance over the distance among phones (edges of the square). The experiments for each distance in each scenario are repeated ten times to reduce random error. The output of this evaluation are curves that indicate the group tracing success rate (average by times of experiments) at different distances for each scenario.

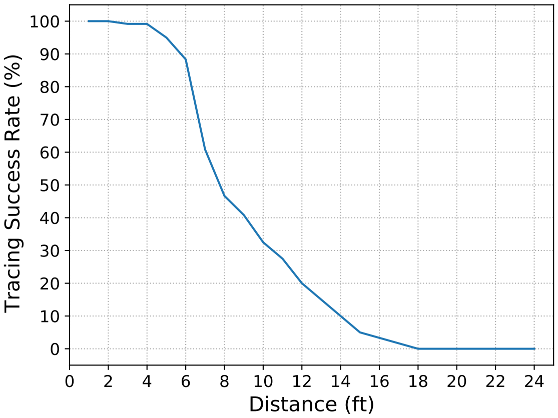
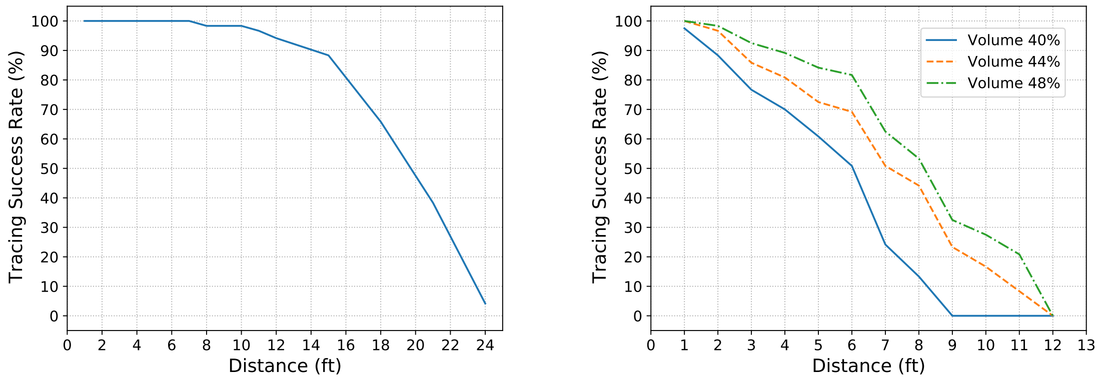


Figure 6: Tracing success rate of 4 mobile phones with different tracing distances under the Noisy In-Pocket scenario. The performance decreases when the distance between people exceeds 6 feet, suggesting ACOUSTIC-TURF's promise to reflect the natural properties of human contact.

Noisy In-Pocket Scenario. This scenario is common in universities or public transportation systems where the environmental noise is loud and everyone has a mobile phone in his/her pocket. In order to simulate the noisy environment, we place a mobile phone at the center of all other phones playing music randomly from Google Play Music at 52% volume. To simulate the in-pocket scenario, each phone is completely covered by a thin fabric. Figure 6 shows the success rate of tracing among 4 phones in the Noisy In-Pocket scenario. The first part of the curve (0 to 5 feet) consistently exceeds 95%, which means that ACOUSTIC-TURF can trace contacts among people who spend more than 10 minutes within 5 feet of each other. The second part of the curve (6 to 12 feet) indicates that ACOUSTIC-TURF's success rate in contact tracking decreases rapidly when the distance between people is larger than 6 feet. The third part of the curve ($\geq$ 13 feet) shows



(a) Full Volume

(b) Non-full Volumes

Figure 7: Communication accuracy of ACOUSTIC-TURF under the Quiet Out-of-Pocket scenario. We show tracking success rates at different volume levels. By adjusting the volume level, ACOUSTIC-TURF scales to various scenarios according to different effective tracing distances.

that ACOUSTIC-TURF is unable to track people who are far from the phone even if their distance remains unchanged for 10 minutes. In conclusion, ACOUSTIC-TURF is a promising solution for contact tracing against COVID-19 in Noisy In-Pocket scenarios when phones broadcast *PRID*s at full volume, as its effective tracing distance is close to 6 feet.

Quiet Out-of-Pocket Scenario. This scenario corresponds to an airport departure gate (an hour before takeoff) or inside a library. We use a similar configuration as in §6.2: 4 phones are placed on a flat hardwood floor at vertices of a square. The difference is that the phone sending background noise and the fabric are removed for the Quiet Out-of-Pocket scenario. Figure 7(a) shows the success rate of tracing between 4 Pixel 4 phones in the Quiet Out-of-Pocket scenario. The curve is constantly at 100% until a distance of 12 feet, which means that in this scenario, ACOUSTIC-TURF can accurately track users within a 12-foot distance if this distance is fixed for at least 10 minutes. After 12 feet, the figure shows a rapid decrease of the success rate. We conclude that the effective tracing distance of ACOUSTIC-TURF in the Quiet Out-of-Pocket scenario is near 12 feet. ACOUSTIC-TURF is not a good fit for contact tracing against COVID-19 in the Quiet Out-of-Pocket scenario when devices broadcast at full volume, as its effective tracing distance greatly exceeds 6 feet.

In Figure 7(a), the effective tracing range is up to 12 feet in the Quiet Out-of-Pocket scenario where phones broadcast at full volume. We evaluate the effective tracing distance under different volume levels, using the same setup with experiments in the Quiet Out-of-Pocket scenario but with varying volume levels. As there are 25 volume levels in the Pixel 4, we divide volume level between mute and full volume using 25 volume levels; hence, each pair of consecutive volume levels differs by 4%. First, we adjust the volume level to determine the minimal volume level that allows ACOUSTIC-TURF to get a reasonable success rate at 6 feet (success rate $\geq$ 50%). Next, we find the minimal volume level that allows ACOUSTIC-TURF to get a reliable success rate at 6 feet (success rate $\geq$ 90%). Afterwards, we explore the tracing success rate curve with each volume level between minimal and maximal volume. Figure 7(b) shows that ACOUSTIC-TURF's tracking success rate using different volume levels in the Quiet Out-of-Pocket scenario. At 48% volume level, the

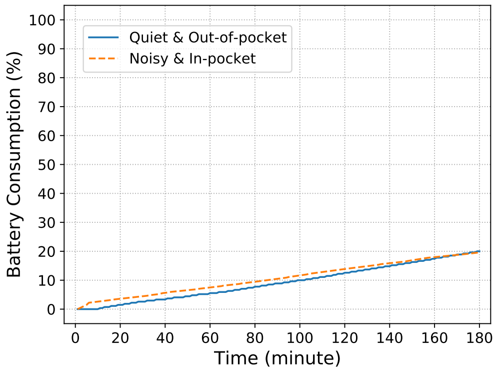


Figure 8: Power consumption of ACOUSTIC-TURF in different scenarios. The overall consumption is satisfactory for practical usage.

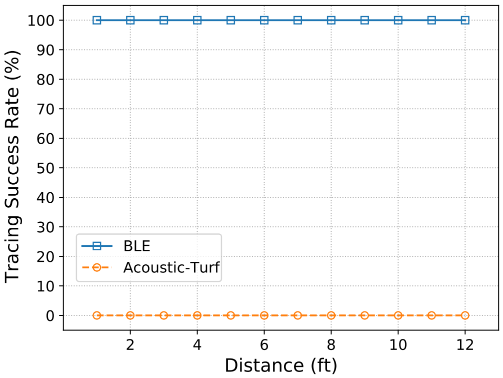


Figure 9: Success rate comparison between ACOUSTIC-TURF and BLE on through-wall communication.

curve in the figure indicates that when all phones broadcast *RPID*s, the turning point of tracking success rate is about 6 feet, which means that the effective tracking distance is at about 6 feet. By adjusting the volume level, it is possible to limit ACOUSTIC-TURF's effective tracking distance near 6 feet. However, since the two scenarios we involved in the evaluation have different "best volume" levels, the devices need to detect ambient environmental noise (in or out of pocket) to determine the volume level used to broadcast RPIDs. This is part of our future work.

Remark. With our evaluation in two scenarios, we conclude that ACOUSTIC-TURF is able to track high-risk contact made between users. We notice, however, that different scenarios require different volume levels to control the effective tracking distance. Therefore, ACOUSTIC-TURF needs to adjust the volume level according to the kind of scenario the devices are in, which forms part of our future work.

6.3 Evaluation of Energy Consumption

To evaluate the battery use of ACOUSTIC-TURF in the two scenarios, we use the same configuration of devices as in §6.2. However, we fix the distance between Pixel 4 devices at exactly 6 feet and have all devices fully charged before start running ACOUSTIC-TURF. All devices keep broadcasting and receiving *RPID* for three hours. We check the battery in the ACOUSTIC-TURF app every 60 seconds to construct Figure 8, which indicates that ACOUSTIC-TURF uses 6% of battery each hour(on average) when both broadcasting and receiving *RPID* in both scenarios at full volume. The battery consumption in the Noisy In-Pocket scenario is slightly higher than that in the Quiet Out-of-Pocket scenario. An explanation for this phenomenon is that in the Noisy In-Pocket scenario, ACOUSTIC-TURF increases the unit number per data frame to ensure the required tracking success rate, which slightly increases power consumption. Hence, we conclude that running ACOUSTIC-TURF in the background does not reduce battery lifetime to under 8 hours, which lets users charge their devices before their batteries are depleted.

6.4 Comparison to Bluetooth-Based Solutions

A major concern about contact tracing using Bluetooth-based solutions is that such wireless signals can permeate through certain obstacles. This means that these solutions regard people separated by a wall as having been in contact with each other, whereas we know this is not the case. In this section, we compare ACOUSTIC-TURF to BLE with a focus on the communication performance when individuals are separated by a wall. Concretely, we put 2 Pixel 4 devices on each side of a wall on the hardwood floor without fabric covers. We control the distance between these devices to ensure they are placed at vertices of a square. Following §6.2, all devices start broadcast and receive *RPID* at the same time for 10 minutes; we collect the number of *RPID*s received by each device to evaluate ACOUSTIC-TURF's ability to permeate the wall. In addition, the devices run a BLE broadcasting app under the same setting. Figure 9 shows the success rates of ACOUSTIC-TURF and BLE. The figure shows that ACOUSTIC-TURF does not track people who are on different sides of a wall, which reflects the natural properties of contact tracing for COVID-19.

7 Discussions and Future Work

Further Design on Acoustic Communications and Random ID Generation. Our current design for acoustic communication illustrates the feasibility of such communication on mobile phones for contact tracing spanning a 6-foot range among multiple users in occluded and noisy environments. Further improvements to our design are possible, including increased reliability and energy efficient communication for the same purpose. In addition, we can extend our design for other applications (such as IoT ones) by improving its throughput over various distances. Our current cryptographic protocol can also be improved in many ways. For instance, we can adopt a similar design as Apple and Google's for generating each day's $DRID$ independently instead of using a hash chain. To improve the precision of matching, we may restrict infected users to upload $RPID$s during certain time windows only (as DP3T [25] proposed) instead of regenerating $RPID$s from $DRID$s. In addition, our system does not keep IDs of infected users secret; we can leverage private set intersection [10] for greater privacy.

More Thorough Implementation and Evaluation. In the above sections, we reported our implementation and evaluation on Android mobile phones. Due to COVID-19, all authors have been staying at home while conducting this research. It is difficult for us to implement ACOUSTIC-TURF on multiple platforms (such as iOS) and conduct more thorough evaluations with various numbers of users in more practical working scenarios. We compare the tracing success rate of ACOUSTIC-TURF in Noisy In-Pocket and Quiet Out-of-Pocket scenarios in §6.2 with 4 mobile phones. The tracing success rate curve decreases at around 6 feet in the Noisy In-Pocket scenario. However, the curves of the Quiet Out-of-Pocket scenario decrease at different ranges depending on the speaker volumes. Similar phenomena occur in scenarios with fewer phones. Automatically adjusting various parameters based on working scenarios to ensure contact tracing ranges are around 6 feet remains an important part of our future research.

Other Improvements. We use ultrasonic frequencies ranging from 17,500 Hz to 22,600 Hz. However, due to BASK modulation, our system generates some audible noise, even using carrier waves above the audible frequency threshold. In future work, we aim to achieve inaudible acoustic communication in practice. Our system uses mobile phones' speakers and microphones to realize acoustic communication for contact tracing. Questions may arise about the impact of applications (e.g., phone calls and music playback) on speaker and microphone usage. Although our preliminary tests on this matter show limited impacts, this remains an area for future research. Acoustic signals offer the potential for sensing finer-grained contact detection. In addition, we can incorporate other sensors on mobile phones (such as compasses) in order to infer face orientations, handshakes, and conversations.

8 Conclusion

This paper proposed ACOUSTIC-TURF, a novel contact tracing system that preserves users' privacy by employing acoustic signals on commercial off-the-shelf (COTS) Android phones. ACOUSTIC-TURF performs automated contact tracing and enables infected users to upload their random identifiers (at their discretion) to public health authorities. ACOUSTIC-TURF realized such functionality using a privacy-preserving cryptographic protocol for security and an acoustic communications protocol. We have implemented ACOUSTIC-TURF on COTS mobile phones. Our experimental evaluation showed the promise of our system for privacy-preserving contact tracing using acoustic signals. In particular, ACOUSTIC-TURF exhibited better performance than Bluetooth-based contact tracing solutions in certain scenarios (such as people on different sides of walls). In our system, there is a clear "turf" effect where contact tracing rates dramatically decrease at certain distances. For instance, with 4 phones and the Noisy In-Pocket working scenario, the tracing rate decreased rapidly beyond a 6-foot range. There are multiple avenues to improve ACOUSTIC-TURF, and our immediate future work is to extend ACOUSTIC-TURF to mobile devices running iOS and evaluate the system's performance on an ensemble of devices running both iOS and Android operating systems.

Acknowledgement

The work was supported in part by the National Science Foundation (NSF) under Grant No. CNS 2028547. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] Mimonah Al Qathrady and Ahmed Helmy. 2017. Improving BLE Distance Estimation and Classification Using TX Power and Machine Learning: A Comparative Analysis. In *Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, 79–83.

[2] Privacy-Preserving Contact Tracing Apple and Google. 2020. https://www.apple.com/covid19/contacttracing/. (Accessed on 04/23/2020).

[3] Bill Bostock. 2020. Experts wary of rushed COVID-19 vaccine after Fauci says 12-18 months - Business Insider. https://www.businessinsider.com/coronavirus-vaccine-quest-18-months-fauci-experts-flag-dangers-testing-2020-4. (Accessed on 04/23/2020).

[4] Ran Canetti, Ari Trachtenberg, and Mayank Varia. 2020. Anonymous Collocation Discovery: Harnessing Privacy to Tame the Coronavirus. *arXiv: Computers and Society* (2020).

[5] Justin Chan, Dean P. Foster, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham M. Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Sudheesh Singanamalla, Jacob E. Sunshine, and Stefano Tessaro. 2020. PACT: Privacy Sensitive Protocols and Mechanisms for Mobile Contact Tracing. *ArXiv* abs/2004.03544 (2020).

[6] OpenTrace Community. 2020. OpenTrace Calibration. Device calibration data and Trial Methodologies for testing implementations of the BlueTrace protocol. https://github.com/opentrace-community/opentrace-calibration. (Accessed on 05/11/2020).

[7] CovidSafe. 2020. https://github.com/CovidSafe. (Accessed on 04/23/2020).

[8] Andrew Crocker, Kurt Opsahl, and Bennett Cyphers. 2020. Bluetooth contact tracing needs bigger, better data. https://www.technologyreview.com/2020/04/22/1000353/bluetooth-contact-tracing-needs-bigger-better-data/. (Accessed on 04/23/2020).

[9] Paresh Dave. 2020. Explainer: How smartphone apps can help 'contact trace' the new coronavirus - Reuters. https://www.reuters.com/article/us-health-coronavirus-tracing-apps-expla/explainer-how-smartphone-apps-can-help-contact-trace-the-new-coronavirus-idUSKCN21W2I8. (Accessed on 04/23/2020).

[10] Emiliano De Cristofaro and Gene Tsudik. 2010. Practical private set intersection protocols with linear complexity. In *International Conference on Financial Cryptography and Data Security*. Springer, Springer, 143–159.

[11] Sidney Fussell and Will Knight. 2020. The Apple-Google Contact Tracing Plan Won't Stop Covid Alone. https://www.wired.com/story/apple-google-contact-tracing-wont-stop-covid-alone/. (Accessed on 04/23/2020).

[12] Po-Shen Loh. [n.d.]. NOVID. https://www.novid.org/. (Accessed on 06/22/2020).

[13] Leonardo Maccari and Valeria Cagno. 2020. Do we need a Contact Tracing App? *ArXiv* abs/2005.10187 (2020).

[14] Rajalakshmi Nandakumar, Krishna Kant Chintalapudi, Venkat Padmanabhan, and Ramarathnam Venkatesan. 2013. Dhwani: secure peer-to-peer acoustic NFC. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 63–74.

[15] Sourabh Niyogi, James Petrie, Scott Leibrand, Jack Gallagher, Manu Eder Hamish, Zsombor Szabo, George Danezis, Ian Miers, Henry de Valence, and Daniel Reusche. 2020. TCNCoalition/TCN: Specification and reference implementation of the TCN Protocol for decentralized, privacy-preserving contact tracing. https://github.com/TCNCoalition/TCN. (Accessed on 04/23/2020).

[16] Dennis Normile. 2020. Coronavirus cases have dropped sharply in South Korea. Whats the secret to its success? https://www.sciencemag.org/news/2020/03/coronavirus-cases-have-dropped-sharply-south-korea-whats-secret-its-success. (Accessed on 04/23/2020).

[17] Ed Novak, Zhuofan Tang, and Qun Li. 2018. Ultrasound proximity networking on smart mobile devices for IoT applications. *IEEE Internet of Things Journal* 6, 1 (2018), 399–409.

[18] Government of Singapore. 2020. Trace Together, safer together. https://www.tracetogether.gov.sg. (Accessed on 04/23/2020).

[19] Android Open Source Project. 2019. AdvertiseSettings.Builder. https://developer.android.com/reference/android/bluetooth/le/AdvertiseSettings.Builder#setTxPowerLevel(int). (Accessed on 05/11/2020).

[20] Android Open Source Project. 2019. ScanRecord. https://developer.android.com/reference/android/bluetooth/le/ScanRecord#getTxPowerLevel(). (Accessed on 05/11/2020).

[21] Android Open Source Project. 2019. ScanResult. https://developer.android.com/reference/android/bluetooth/le/ScanResult#getRssi(). (Accessed on 05/11/2020).

[22] Ronald L. Rivest, Jon Callas, Ran Canetti, Kevin Esvelt, Daniel Kahn Gillmor, Yael Tauman Kalai, Anna Lysyanskaya, Adam Norige, Ramesh Raskar, Adi Shamir, Emily Shen, Israel Soibelman, Michael Specter, Vanessa Teague, Ari Trachtenberg, Mayank Varia, Marc Viera, Daniel Weitzner, John Wilkinson, and Marc Zissman. 2020. The PACT protocol specification. https://pact.mit.edu/wp-content/uploads/2020/04/The-PACT-protocol-specification-ver-0.1.pdf. (Accessed on 04/23/2020).

[23] ROBERT ROBust and privacy-presERving proximity Tracing protocol. 2020. https://github.com/ROBERT-proximity-tracing. (Accessed on 05/12/2020).

[24] G. E. Santagati and T. Melodia. 2017. A Software-Defined Ultrasonic Networking Framework for Wearable Devices. *IEEE/ACM Transactions on Networking* 25, 2, 960–973.

[25] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, et al. 2020. Decentralized Privacy-Preserving Proximity Tracing. https://github.com/DP3T/documents. (Accessed on 04/23/2020).

[26] Tina White, Rhys Fenwick, Isaiah Becker-Mayer, James Petrie, Zsombor Szabo, Daniel Blank, Jesse Colligan, Mike Hittle, Mark Ingle, Oliver Nash, Victoria Nguyen, Jeff Schwaber, Akhil Veeraghanta, Mikhail Voloshin, Sydney Von Arx, and Helen Xue. 2020. Slowing the spread of infectious diseases using crowdsourced data. https://www.covid-watch.org/article. (Accessed on 04/23/2020).

[27] Faheem Zafari, Athanasios Gkelias, and Kin K Leung. 2019. A survey of indoor localization systems and technologies. *IEEE Communications Surveys & Tutorials* 21, 3 (2019), 2568–2599.

[28] Huanle Zhang, Wan Du, Pengfei Zhou, Mo Li, and Prasant Mohapatra. 2017. An acoustic-based encounter profiling system. *IEEE Transactions on Mobile Computing* 17, 8 (2017), 1750–1763.