Linking Bluetooth LE & Classic and Implications for Privacy-Preserving Bluetooth-Based Protocols

Norbert Ludant
Northeastern University
Boston, USA
ludant.n@northeastern.edu

Tien D. Vo-Huu
Northeastern University
Boston, USA
vo-huu.d@northeastern.edu

Sashank Narain
University of Massachusetts Lowell
Lowell, USA
sashank_narain@uml.edu

Guevara Noubir
Northeastern University
Boston, USA
g.noubir@northeastern.edu

***Abstract*—Bluetooth Low Energy advertisements are increasingly used for proximity privacy-preserving protocols. We investigate information leakage from BLE advertisements. Our analysis, among other things, reveals that the design of today's Bluetooth chips enables the linking of BLE advertisements to Bluetooth Classic (BTC) frames, and to a globally unique identifier (BDADDR). We demonstrate that the inference of the BDADDR from BLE advertisements is robust achieving over 90% reliability across apps, mobile devices, density of devices, and tens of meters away from the victims. We discuss the implications of current chipsets vulnerability on privacy-preserving protocols. The attack, for instance, reveals the BDADDR of devices of infected users of contact-tracing apps. We also discuss how the vulnerability can lead to de-anonymization of victims. Furthermore, current mobile devices do not allow selective disabling of BTC independently of BLE which renders simple countermeasures impractical. We developed several mitigations for the Android OS and the Bluetooth stack and demonstrate their efficacy.**

I. Introduction

Mobile and wireless systems have revolutionized how we access and share information. However, this success has come with increased concerns about user privacy. For instance, users are increasingly aware and concerned about the implications of disclosure of location information [1]. Similarly, governments and legislators have increased their scrutiny as illustrated by the US Congress Location Privacy Protection Act [2].

Protecting users privacy is known to be challenging as private information can leak through numerous channels. As such, protections should take into account a variety of adversarial models. Today, most wireless communications systems have adopted security protocols with encryption, integrity protection, and even forward security features. However, the *linkability* of communications remains an important challenge. Linkability of communications can not only enable tracking, but also used as a crucial stepping stone for more sophisticated attacks. For instance, *once a single (linkable) message reveals a private information about a user, it can then be associated with the device*, its mobility patterns, and other linked-information collected or inferred by an adversary.

Over the years, wireless systems increased their defenses against linkability. Cellular systems as early as GSM introduced the Temporary Mobile Subscriber Identifier (TMSI) to minimize leakage of the globally unique and permanent International Mobile Subscriber Identifier (IMSI). Later generations introduced further protections leading to the 5G design not sending permanent identifiers in the clear and instead relying on the Subscriber Concealed Identifier (SUCI) consisting of the permanent SUPI encrypted (IND-CPA) with the public key of the cellular operator [3]. Mobile platforms also recognized the risks associated with linkability and potential attacks. For instance, Android, and iOS rely on Wi-Fi MAC address randomization to prevent leakage and continue to improve their security in light of discovered vulnerabilities [4].

We investigate and demonstrate the linkability of BLE advertisements to a global identifier (BDADDR) that permits both the tracking of BLE users, and linking of their device to any information learned directly or indirectly from advertisements. For instance, in the context of contact tracing apps, the adversary learns the BDADDR of infected users devices. We also discuss certain scenarios when the de-anonymization of such users is possible. Our analysis reveals that the design of today's Bluetooth chips enables (1) the linking between BLE advertisement frames emanating from a mobile device, and (2) more critically, the linking of BLE advertisements to Bluetooth Classic (BTC) frames. This Bluetooth stack vulnerability is fundamental because both BLE and BTC are driven by the same clock with a timeslot duration of $625\mu s$. A detailed analysis enables us to develop and demonstrate an inference technique that links BLE advertisements to devices globally unique BDADDR with over 90% reliability across apps, mobile devices, density of devices, and tens of meters away from the victims. Furthermore, the fact that current mobile devices do not allow selective disabling of BTC independently of BLE renders simple countermeasures impractical. We developed several mitigations for Android OS and Bluetooth stack and demonstrate their efficacy. In summary, we:

- Discovered a linking vulnerability in the design of Bluetooth chips, that leads to attacks on privacy-preserving protocols relying on BLE advertisements.
- Analyze the performance of the attack on Android and iOS devices, in terms of accuracy and speed considering different apps, devices, density of neighboring devices, distance to target, and types of existing Bluetooth traffic.
- Our analysis indicates that the attack is robust. For instance at a distance of 10 meters, an attacker can link BLE advertisements to the globally unique BDADDR

with 77% probability within 1 second of sniffing and with 100% probability within 10 seconds of sniffing.
- We show the implications on privacy-preserving protocols (e.g., linking transient public keys in Apple Find My to BDADDR, or exposing the BDADDR of infected users in contact tracing) and discuss scenarios that can lead to de-anonymization.
- We devise and implement several mitigation techniques for Android to limit the attack's potential. We plan to open source all of our code.
- We responsibly disclosed the attack and proposed mitigations to Google, Apple, and contact tracing systems designers. Google assigned severity S2 to the vulnerability, and Apple is evaluating the impact of this attack.

II. Background: Bluetooth and BLE Privacy-Preserving Protocols

Bluetooth protocols were developed over two decades leading to several backward compatible standards. We focus on the latest configuration supported by a vast majority of mobile devices, the Basic Rate (BR) and Low Energy (LE) combined core configuration [5]. A BR system includes optional Enhanced Data Rate (EDR) and Alternate Media Access Control and Physical (AMP) layer extensions. These protocols are typically referred to as Bluetooth Classic (BTC). Bluetooth Low Energy (BLE) on the other hand aims at minimizing energy consumption. These modes serve unique needs, optimising for energy, bit-rate, and range. However, their physical and MAC layers share common characteristics that prove key to the cross-mode side channel vulnerability. We first provide an overview of the two prominent modes, BTC and BLE, focusing on the mechanisms enabling the attack.

A. Bluetooth Classic (BTC) Overview

BTC operates in the 2.4GHz ISM band, spanning 80MHz. The physical layer uses a Gaussian Frequency Shift Keying (GFSK) modulation (BR mode), and $\pi/4$-DQPSK or 8DPSK (EDR mode). The modulated signal is transmitted at a symbol rate of 1Msym/s, thus providing a bit-rate of 1Mbps (BR mode) and 2Mbps or 3Mbps (EDR mode). Signals are transmitted in 79 relatively narrow band channel spaced by 1MHz. The center frequency ranges from 2.4GHz to 2.4835GHz. The channels are determined by their center frequency $f_k = (2402 + k)$ MHz, $k \in [0, 78]$ where k is the channel index. A channel is divided into slots of $625\mu s$. BTC uses frequency hopping to mitigate interference and selective fading. A BTC packet might last 1 to 5 timeslots but always starts in a synchronized way with the $625\mu s$ period (see Figure 1).

BTC communicating devices form a network topology called piconet, consisting of a master and up to seven slaves. The master determines the channel hopping pattern and time slots schedule. Therefore, time synchronization is required at the initialization stage when two devices initiate a connection. Slaves apply an offset to align their local clock to the master clock. The channel hopping pattern is determined by the master's clock and Bluetooth Device Address BDADDR.

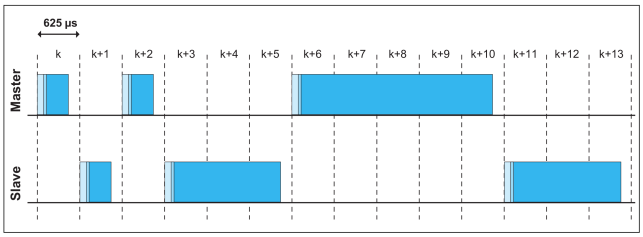


Fig. 1: Bluetooth Classic transmissions using TDD Scheme (src. [5]). Packets can span multiple slots but are always synchronized to a $625\mu s$ period set by the master.

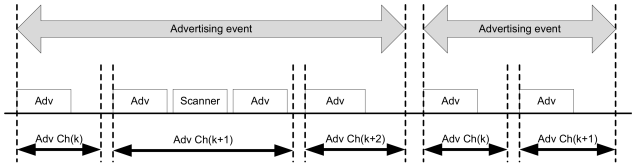


Fig. 2: BLE advertising events over multiple advertisement channels (src. [5]).

A BDADDR consists of a Lower-Address-Part (LAP), a Upper-Address-Part (UAP), and a Non-significant-Address-Part (NAP). The LAP of the master device is used to derive the access code for every packet in the piconet and can be extracted from the header during the frame synchronization stage. The UAP of the master device is used to initialize the generation of Header Error Check (HEC), which is whitened together with the rest of the header. As the UAP is not explicitly transmitted and combined with frequency hopping over 79 channels, BTC was considered to be secure in terms of not revealing devices MAC address. However, this has been recently shown not to be true [6]. The NAP, as its name suggests, does not play an important role in the packet.

B. Bluetooth Low Energy (BLE) Overview

BLE operates over the same spectrum as BTC and has a similar physical layer but optimized for energy. BLE has two modulation schemes, using GFSK: LE 1M and LE Coded with 1Msym/s, and LE 2M with 2Msym/s. BLE supports data bit rates of 1Mbps (LE 1M), 500 kbps and 125kbps (LE Coded), and 2Mbps (LE 2M). The spectrum $[2.4 - 2.4835]$GHz is divided into 40 RF channels with 2MHz spacing. The channels center frequencies are defined as $f_k = (2402 + k \times 2)$ MHz, $k \in [0, 39]$. There are two categories of channels (1) primary advertising (37, 38, 39) and general purpose $(0, \cdots, 36)$.

Unlike BTC, the multi-access scheme of BLE divides time into units called events. There are five types of events, *Advertising, Extended Advertising, Periodic Advertising, Connection* and *Isochronous*. We focus on the advertising events for linking BLE to BTC. Advertising events are an important feature specific to BLE. By broadcasting an advertising packet, a BLE device announces its offering of a service. In this scheme, the advertising device is called advertiser and listening devices are scanners. The scanners either ignore or react to advertisements. An advertising event is defined as a time

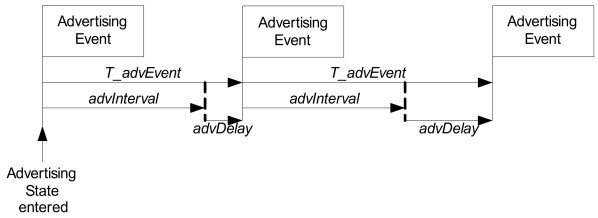


Fig. 3: Illustration of BLE advertising events timing (src. [5]).

window where advertising packets and their responses occur. Figure 2 illustrates advertising packets in the same advertising events with the same information transmitted on different advertising channels. The first advertising packet in an advertising event is also the start of the event. The time between two advertising events is composed of a selected fixed duration, called advertising interval and is a multiple of 625μs in range 20ms to 10485.759375s; and a delay, called advertising delay which is random in a range of 0ms to 10ms to avoid collision among advertisements from different advertisers. Figure 3 shows the timing spacing between advertising events.

BLE incorporates a MAC address randomization scheme, triggered after a period of time or whenever a new advertising or connection is initialized. The standard recommends that a random MAC address is re-generated every 15 minutes. This in principle improves the protection of BLE against tracking.

C. BLE-Based Privacy-Preserving Protocols

We focus on privacy-preserving protocols like Contact Tracing and Apple Find My, because these protocols explicitly rely on the unlinkability (randomization) of BLE advertisements.

Contact Tracing Apps: Digital contact tracing serves an important role in the current COVID-19 pandemic for tracking the spread of the virus [7]. Concerns about privacy led to several initiatives to design privacy-preserving contact tracing systems. Apple and Google teamed-up to design and embed an "Exposure Notification" protocol in iOS and Android platforms [8]. Around the world, other groups proposed precursor designs, alternatives, and complementing solutions, including university consortia and research labs (e.g., DP-3T [9], PACT [10], [11] and PACT [12]). A common privacy goal of these solutions is to *dissociate any information that might connect a smartphone owner's device to the contact tracing information*. Several studies investigated privacy concerns arising from contact tracing apps [13]–[18] and some additional protections were also proposed [16], [19]. Several other contact tracing apps (e.g., Immuni [20], SwissCovid [21], COVID Alert [22], California COVID Notify [23]) were developed using the Exposure Notification Service.

The Exposure Notification service provides privacy through the use of non-personal cryptographically generated identifiers. Each user is assigned a random 16 byte cryptographic key called the Temporary Exposure Key (TEK). This key is setup to roll every day for every user of the system. The TEK is used to generate two identifiers called Rolling Proximity Identifier (RPI) and Associated Encrypted Metadata (AEM) by encrypting the epoch time and a constant using the AES algorithm. In case of the AEM, the RPI is also encrypted to link the identifiers together. The generated RPI and AEM are broadcast to devices in proximity using BLE advertisements. Another technique that the service utilizes to preserve privacy is generating new RPI and AEM values whenever the advertiser's BLE MAC address is randomized. *This way, an attacker is unable to link the identifiers with a device.* The TEK, RPI, and AEM are all required to determine whether a user was in contact with an infected user. The infected user uploads their last 14 TEKs to a server. These TEKs (called Diagnosis Keys) are combined and downloaded on all devices and then used to generate RPIs and AEMs. A user is alerted of possible contact if their device contains a minimum count of infected RPIs and AEMs. We refer the readers to the design documentation for a detailed description [8]. Although other apps use other cryptographic protocols and algorithms, they all use the same underlying Android BLE implementation for communication, the subject of our analysis.

Apple Find My: Apple introduced the Find My feature to provide customers a tool to find their devices in a privacy-preserving manner [24], [25]. To perform its intended function, the protocol requires the user to own two Apple devices that share a private key known only to those devices. Corresponding to the private key, the devices generate transient public keys that are broadcast over BLE advertisements. Both private and public keys do not reveal any information about the device owner. In order to mitigate linkability, these public keys change whenever the advertiser's BLE MAC address is randomized. The transient public keys broadcast using BLE advertisements are received by other devices in proximity which use their own location, encrypt the location using the received public key, and then upload the encrypted location and public key's hash value to the cloud. When a user realizes that their device is missing, they use the other device to upload the hash values of public keys as identifiers and download all the encrypted locations corresponding to those hash values. These encrypted locations are then decrypted using the private key to reveal the location of the lost device.

III. System and Attacker Model

A. System Model: The Bluetooth (BTC+BLE) Stack

We consider scenarios that are common among smartphone users. Bluetooth is embedded in virtually every phone, car, laptop, mouse, keyboard, game console, and wearable device (billions of devices ship every year [26]). Its services range from communication with peripherals, audio and video streaming, and even transmission of health information (e.g., fitness trackers). We assume a smartphone user that regularly uses Bluetooth on their device for such services. For the purpose of the attack, it does not matter how the services are used, what data is exchanged, if a session is active, which apps are used, or the mode of transmission.

Current Android and iOS devices support both BTC and BLE. BLE is used by many services for data transmission and advertisements. Examples of some services include Samsung

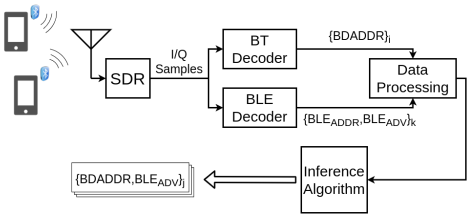


Fig. 4: Flow Diagram for the Attack System.

Health, Apple Find My, Apple iBeacon, and Apple/Google Exposure Notification (contact-tracing). We focus on BLE advertisements, the communication medium used by the examples above. We assume the advertisements are running in the background. This is now a common scenario for millions of users that run the aforementioned apps on their smartphones as these apps start broadcasting as soon as the device is booted and keep broadcasting at all times.

B. Attacker Model: Abusing BTC+BLE Timing Information

We assume an attacker in proximity to the victim(s). The attacker is able to record and process RF (I&Q) samples in the 2.4GHz band. This can be achieved using a software defined radio (SDR) or even mobile phones supporting the Nexmon framework [27]. We demonstrate that the attacker can recover relevant BTC and BLE packets, their timing, and link them to the unique BDADDR along with any additional information extracted from such packets. This might include public keys for Apple Find My, RPIs from Exposure Notification, and user identifiers [28]. We also assume that attacker is able to retrieve public information for each of the considered systems. In the case of Exposure Notification, this includes the published Diagnostic Keys of infected users. For Apple Find My, it would include the encrypted location given the device randomized public key hash. The attacker is passive and does not inject wireless signals or interact with the victim.

IV. Linkage and Tracking: Attack Overview

We first provide an overview of the attack approach, then discuss the characteristics of the side-channel linking BTC and BLE. Through a set of experiments, we demonstrate the existence of a dependency, although noisy, between transmission times. We then summarize the challenges that need to be investigated to understand the full potential of the attack.

A. Overview of the Attack

The goal of the attacker is to (1) link BLE advertisements and BTC transmissions, (2) derive globally unique identifier from BTC transmissions (BDADDR), and (3) use the link for tracking BLE users and for linking their device to any information learned directly or indirectly from the advertisements. To achieve this goal, the attacker needs to handle multiple steps - from the correct reception of RF emissions, to extracting timing information and finally discerning which transmissions are coming from the same device. The key components and flow of the attack is outlined in Figure 4.

First, our attack system decodes the RF samples to simultaneously extract information of both BTC and BLE transmissions. For BTC, this includes the global BDADDR and the timestamp of each packet received. For BLE, the information includes the BLE advertisements (BLEADV) and their timestamps. The collected information is then processed to derive a timing relation between each pair of BDADDR and BLEADV. The timing relation, in this context, is the alignment quantification between the clock of BTC and the clock of BLE. Intuitively, if a device supports both BTC and BLE, their modules are combined in the same Bluetooth chip and their time reference is derived from the same clock source.

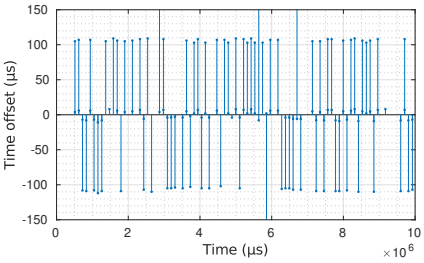
The inference algorithm is the core of the attack. It computes the time alignment between BTC and BLE transmissions addressing the peculiarities of BLE as described in section IV-D, such as irregular channel offsets. We first show preliminary results in Section IV-B, and delve into the details of the attack in Section V.

B. Coupling of BTC and BLE Advertisements

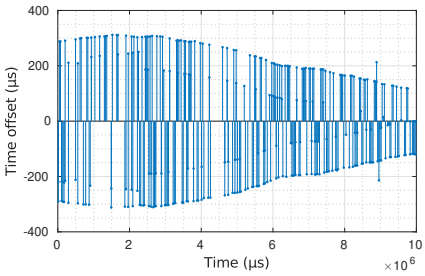
The key premise of the attack is that BLE advertisements can be linked to BTC transmissions, which themselves reveal the device BDADDR. Our initial guess was that given the similar frequency-hopping pattern of BTC and BLE (1,600 hops per second) and the high integration of Bluetooth SoCs, it is likely that both radios are driven by a common clock. We conducted a set of basic experiments to determine the nature of coupling between BTC and BLE advertisements.

The results of our preliminary analysis are shown in Figure 5. Details of the experimental setup are in Section VI. These results compare the time alignment of two different sets of BLE advertisements with the same set of BTC transmissions. The first set of BLE advertisements are transmitted from the same device as the BTC transmissions (our test device), while the second set corresponds to the BLE advertisements of another device. Figure 5a shows the alignment of BTC transmissions with BLE originating from the same device (modulo $625\mu s$), while Figure 5b shows the alignment of second BLE set from other device. This result confirmed our intuition of coupling between BLE advertisements and BTC emissions (i.e., either fixed *low* offset or *fixed* offset around $\pm 110\mu s$). In Section V, we show that these offsets result from irregular transmissions, scanning request-respond procedure, channel impairments, etc. The offset to the other device is *non-constant*. Yet, it is not trivial to infer the coupling since there are multiple fixed-offset patterns (two in Figure 5a). As we do not have visibility into driver implementations, we have not been able to confirm our intuition for the reason of these offset patterns. We leave the detailed analysis of this artifact for future work.

Another observation during preliminary analysis was that even when BLE MAC randomization occurred, the timing of advertising events remained the same since the randomization does not affect the baseband clock of the Bluetooth module. This makes it possible to link BLE advertisements pre- and post- randomization by computing the alignment between BLE



(a) Time offset between BLE advertisements of test device with the BTC transmissions from the same device.



(b) Time offset between BLE advertisements of test device with the BTC transmissions of another device.

Fig. 5: Example of BLE advertisement coupling with BTC transmissions from the same device (left) and with different device (right).

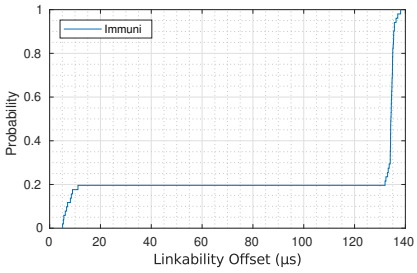


Fig. 6: Linkability ecdf RMSE

advertisements, instead of linking them to BTC transmissions. As such, this attack could potentially be used to link BLE advertisements over long periods of time even in the absence of BTC emissions.

These preliminary results support our hypothesis, and indicate the necessity of an in-depth analysis for a robust attack.

C. Reliability of a Simple End-to-End Attack

In light of our early analysis, we developed a basic end-to-end attack for linking BTC and BLE transmissions in scenarios without any knowledge about a specific target device. Our only assumption is that the target is transmitting BLE advertisements along with an active BTC connection. We note that this is a realistic attack scenario as many apps such as Exposure Notification, Apple Find My and iBeacon rely on BLE advertisements, while users still rely on BTC for streaming music and videos to their headsets. As such, the two may often be used together enabling the attack.

Analyzing 10 seconds of samples, we report on the Root Mean Square Error of offset between BLE and BTC in Figure 6. We link BTC transmissions and BLE advertisements from Immuni [20], a contact tracing app, only 20% of the time (using a RMSE threshold of 13). These initial results indicate not only the possibility of a robust attack but also the need for a systematic analysis accounting for various nuances of BTC/BLE, Android Bluetooth Stack, mobile apps, etc.

D. Challenges Impacting Attack Performance

The challenges are not only due to the nature of a sniffing attack at the physical layer, but also the complexity of Bluetooth protocols and heterogeneity of hardware implementations.

Wireless impairment. The attack relies on sniffing and decoding RF emissions. It is therefore limited by the channel impairments, such as the impact of interference and distance to the target devices. These limitations are particularly significant for BLE advertisements, especially in the case of Exposure Notification which is setup to transmit at low power. In crowded scenarios with multiple wireless devices, as the interference and collisions are unavoidable, the quality of our sniffing is significantly impacted.

Unexpected timing offset pattern. The coupling of BLE advertisements and BTC transmissions appear as a constant offset but not always small. Furthermore, the timing offsets are a combination of multiple patterns resulting in the challenge of distinguishing between a bias misalignment, which can be compensated for, and a real misalignment of a non-coupling.

Tight coupling of BTC and BLE in Android. The Android OS provides no automated mechanism to disable BTC even when all paired devices are disconnected. Furthermore, the OS couples BTC and BLE together such that apps must enable BTC on the device if they need to use BLE advertisements. The above scenarios increases the possibility of a device to reconnect with a previously paired device even when not required. The OS provides no notifications of concurrent BTC and BLE transmissions making mitigations more challenging.

E. Privacy Implications of the Attack

This vulnerability enables an attacker to link information received via BLE advertisements to a device BDADDR. This breaks a key assumption of privacy-preserving BLE-based protocols as most explicitly and critically rely on the unlinkability (randomization) of BLE advertisements.

In case of the Exposure Notification service, the BLE advertisement data consists of the $RPIs$ broadcast by the user devices. These $RPIs$ are generated such that they do not reveal the user's identity. However, the vulnerability we expose will link them to the BDADDR. An adversary (or coalition) that periodically downloads the Diagnosis keys, can then generate the $RPIs$ of *infected* users and compare them to $RPIs$ previously recorded and linked to BDADDRs. This does not fully de-anonymize the infected users. However, it can serve as a stepping stone to full de-anonymization of infected users when combined with any other side-channel or attack that links the BDADDR to users private information. For instance, on Android, system apps can already access the BDADDR of the device, Bluetooth multi-player games can access the BDADDR of paired devices, as can paired IoT devices such as smart speakers. These apps can potentially link the BDADDR to any information they have about the user. Recent works [28], [29] demonstrated that other BLE-based Apple protocols such as AWDL, AirDrop and Nearby

Action can reveal user identifiers such as hostnames, email hashes and phone number hashes. In case of hostname, the study revealed that about 75% of these hostnames contain the user's first or last name or both. With sufficient incentives, it is imaginable that an underground market could emerge for linking BDADDR to user identities especially because it is now possible to track users based on their BDADDR [6].

In case of the Apple Find My service, the BLE advertisements broadcast transient public keys that are not tied to a user. Since the only information retrievable (by an adversary who knows the transient public key hash) from Apple's servers is the location encrypted with the transient key, it is harder to imagine powerful attacks that leverage the BTC-BLE linkage vulnerability. One possible attack assuming an adversary who knows the BDADDR of a target user (as discussed above) is that he can then link the public keys to the user identifier. An adversary (or coalition) can use this knowledge to launch a targeted attack where they first steal the user's device, and then start spoofing their location and encrypting the spoofed location using the user's public keys. This can cause the user to identify the lost device location at a spoofed location of the attacker(s) choice, leading to physical threats to the user.

Given that the BTC-BLE linkage vulnerability appears to have more severe impact on the Exposure Notification protocols, we will focus mostly on such applications.

V. The Detailed Attack

A. Analyzing BLE and BTC Coupling

We first analyze the nature of the coupling between BLE and BTC. We setup a Samsung Galaxy S6 phone to stream audio to a Bluetooth headset over BTC, while broadcasting BLE advertisements using a mobile app. The app called Immuni uses Exposure Notification service as its underlying BLE advertiser. Immuni is the official exposure notification app of the Italian government and is used by over one million users. For this part, we posit that Immuni is representative of many contact tracing apps. We note that this experiment was performed in an environment with proximity to several other devices using BTC communication and BLE advertisements. This enabled us to also determine whether there could be other devices with clocks aligned with our test device.

Our analysis follows the attacker model of Section III-B. We built an integrated and synchronized SDR-based BLE decoder [30] with BTC decoder [6]. We use a USRP B210 SDR to sample half of the Bluetooth spectrum. First, we record all BTC transmissions and BLE advertisements over a 10s interval. To protect the privacy of individuals in our environment, we record only timestamps, BLE randomized address and service UUID, and discard everything else. From the recordings, we calculated the time difference between each BLE advertisement and the closest BTC transmissions (modulo $625\ \mu s$). This time difference is used to determine whether a BTC transmission and BLE advertisement are from the same device, specifically our test device. A low time difference indicates a higher probability of linkage between same device BTC and BLE transmissions. Similarly, a higher difference may indicate that the BTC transmissions and BLE advertisements are from different devices.

To analyze the coupling, we collect 10 second recordings every minute for a 1 hour period. Our goal was to determine if the attack is achievable and repeatable over extended periods of time. The attack system is implemented following the flow graph of Figure 4. We compute the time alignment between each BLE advertisement and BTC transmission. In order to have a single metric to determine how well linked -or not- BTC and BLE transmissions are, we compute the Root Mean Square Error (RMSE) of the misalignment values for each recording. A linkability threshold is set to determine the maximum misalignment per packet.

Irregular transmissions. As described in Section II-B, a BLE advertiser should transmit the same advertisement over all three channels (37, 38 and 39) within an advertisement event. We found the transmissions over the first channel typically aligned to BTC transmissions from the same device. However, subsequent transmissions within the event are not necessarily aligned to the BTC transmissions. In other words, if advertisements within an event are sent over channels N, $N+1$ and $N+2$ in that order, the transmission on channel N will be aligned to BTC transmissions from the same device, but $N+1$ and $N+2$ will see an *offset* to channel N. Furthermore, in our initial experiments on several Android devices, we observed a *second* abnormality. Frequently transmissions on a specific channel would not occur, but the remaining channels would still transmit an advertisement. This irregular behavior was what led to the low performance for the attack described in Section IV-C. The behavior mandates that a robust attack algorithm should not rely on a specific channel and fixed offset but should adapt to account for such variability. To rule out problems with our SDR setup, we implemented a simple BLE scanner app for Android to monitor advertisements. The scanner app confirmed the irregular behavior that advertisements are sometimes not transmitted over specific channels for extended periods, up to 10 seconds.

Unexpected channel offset. In a BLE advertising session, subsequent advertisements transmitted over the same channel are typically aligned for the entire duration of the session. As such, the time offset from channel N to $N+1$ or to $N+2$ remains constant. However, there are cases when this is not the case. An example is be the presence of a connectable or scannable flag in the advertisement (e.g., type ADV_IND, ADV_SCAN_IND). Such flags trigger a response after transmitting the advertisement on channel N which can cause a delay in advertising packets on subsequent channels. Furthermore, the separation between channels N, $N+1$ and $N+2$ can vary depending on the device as well as the advertisement configuration. Analyzing the channel offset for a set of devices, we found a varying interval between advertisement packets. Even for the same app, we measured separations between channels 37 and 38 of 380us and 1.25ms for two different devices. Moreover, the advertising interval is affected by the choice of advertising app.

B. Linking Same Device BTC-BLE Transmissions

To link a BLE advertisement to the BTC BDADDR, we need to quantify the misalignment of corresponding packets timestamps. This quantity is computed as the time difference between each BLE advertisement and the nearest BTC transmission (modulo $625\mu s$). The modulus $625\mu s$ is in fact the time unit defined in BTC, to which all BTC transmissions in a piconet are aligned to. In BLE, the advertisements instead align to advertising events with random intervals. Despite different schedule procedures in BLE and BTC, our techniques still infer the alignment and reveal the linking. For ease of presentation, we describe the attack for a window of samples:

1) Record I&Q RF samples of considered spectrum.
2) Process signal and decode to reveal N BTC transmitters $\{BTC_i\}, i \in [0, N-1]$ and $M \times K$ BLE advertisements $\{BLE_{j,k}\}$, $j \in [0, M-1]$, $k \in [0, K-1]$, where M and K are the number of different advertisements and number of different advertising channel indices respectively.
3) For each combination of BTC_i and $BLE_{j,k}$, compute the time distance (modulo $625\mu s$) $\{D^{(l)}_{i,j,k}\}$, $l \in [0, L-1]$ for each of L $BLE_{j,k}$ advertising packets and the nearest BTC_i packets then derives the combination's score as:
   - $S_{i,j,k} = \sqrt{\frac{1}{L}\sum_{l=0}^{L-1}(D^{(l)}_{i,j,k})^2}$ if $k = 0$
   - $S_{i,j,k} = \sqrt{\frac{1}{L}\sum_{l=0}^{L-1}(D^{(l)}_{i,j,k} - mode(\{D_{i,j,k}\}))^2}$ if $k > 0$
4) The attacker links BTC_i and BLE_i if $(S_{i,j,k} < \bar{S}$ & $L_k > \bar{L})$, where $\bar{S}$ and $\bar{L}$ are the thresholds of score and of number of advertising packets respectively.

Consider an attacker performing a linking attack against N BTC devices and M BLE Advertisements. In this scenario, each BTC device is a master in a connection with a slave. The master devices, as combined BTC-BLE cores, broadcast M BLE advertisements in parallel with BTC transmissions.

First, the attacker starts a recording of the wireless environment to capture signals emitted by the Bluetooth devices. Then, process the recording to derive information such as BTC BDADDRs and timing of its packets, and BLE Advertisements (MAC address) and timing of associating packets. The attacker now has two sets of data, one of N BTC devices $\{BTC_i\}$ and the other of BLE advertisements $\{BLE_j\}$.

Second, the attacker analyzes the coupling of each pair of BTC device and BLE advertisement. To analyze the coupling of a pair, we use the time difference (modulo $625\mu s$) between each BLE advertising packet and the nearest BTC packet. This searching for the nearest is limited to a window of time around the examining BLE advertising packet to prevent possible effect of the drift in the device's Bluetooth clock. If there is no BTC packet found in this window, we discard the BLE advertising packet. Since each advertisement can be transmitted on multiple channels, we consider the timing on each channel separately and denote $k \in [0, K-1]$ as channel index of K advertising channels in use. At the end of this stage, for each pair of BTC device BTC_i and BLE advertisement $BLE_{j,k}$, we obtain a vector of L elements timing offset values $\{D^{(l)}_{i,j,k}\}, l \in [0, L-1]$ for each BLE advertising packet on each channel to nearest BTC packet.

In the last stage, we quantify the coupling by computing a score $S_{i,j,k} = \sqrt{\frac{1}{L}\sum_{l=0}^{L-1}(D^{(l)}_{i,j,k})^2}$ as the Root-Mean-Square-Error of the timing offsets for each pair of BTC_i and $BLE_{j,k}$. Lower score correlates with likely linkability. A linking decision is made based on experimental thresholds as 1) if $S_{i,j,k} < \bar{S}$, where $\bar{S}$ is a score threshold, and 2) if $L_k > \bar{L}$, where L_k is the number of BLE advertising packets and $\bar{L}$ is its required amount. From our measurements, we find that an experimental threshold of 10 μs is able to clearly delimit transmissions originating from one device.

As stated in the previous subsection, we find that channels following the initial channel are not aligned to 625 μs, a priori limiting the use of channels other than the initial one. Nonetheless, we find that although the offset on subsequent channels appears to be device and advertisement specific, this offset is constant for the same advertisement. This fact makes it possible to align channels $k+1$ and $k+2$ to 625 by computing and substracting the most repeated offset value -the mode- for a given advertisement. Transmissions coming from other devices do not have a constant offset, as they drift over time (fig. 5). In this way, we are able to rely on channels other than the initial one for our linkability attack.

We also observed abnormal offset values caused by scanning procedures triggered within an advertisement event. Due to the asynchronous nature of the scanning, an advertising packet on the second and third advertising channels can be delayed by a specific or random amount of time depending on the manufacturer's implementation, adding unexpected values to the timing offset on these channels. As the actual value of the added offset is not possible to forecast, we do not rely on measurements from an advertisement event that had a scanning procedure triggered.

VI. Experimental Evaluation Setup

The goal of our measurements is to understand the potential of the vulnerabilities. We attempt to answer questions such as whether devices with different chipsets are vulnerable to the attack, feasibility across multiple apps, as well as impact of range, density of devices, and existence of a BTC connection.

For our experiments, we use a low-cost USRP B210 SDR to record a portion of the Bluetooth RF spectrum. The USRP B210 is able to cover 56 MHz of the target RF bandwidth. We sample the lower 44 MHz of the BT spectrum. In this way, measurements with the B210 board include half the BTC bandwidth and channels 37 (2402 MHz) and 38 (2426 MHz) of BLE advertisements. Note that this does not limit the attack, as advertising apps broadcast the same information on channels 37, 38 and 39. We do not record the whole bandwidth as it would require two B210 or more expensive SDRs. Later, we show this setup sufficient for highly accurate attacks. The SDR records the RF spectrum in the form of I&Q samples that it relays to the host computer over USB 3.0.

To evaluate the effect on multiple smartphone models in differing scenarios, we chose the following devices for our experiments - Samsung Galaxy S6 (SGS6), Pixel 3A, Xiaomi

Mi8, Motorola Moto G5 Plus and iPhone X. These devices encompass different chipsets, operating systems and hardware drivers. Along with the devices, we explore whether only specific apps and/or advertisement configurations are affected by the vulnerability or whether it is fundamental to all BLE advertisements. We select a set of apps to answer this question. Our baseline app was nRF Connect that directly uses the Bluetooth API and permits configuring advertisements parameters such as transmit power and latency. Secondly, we choose an app that uses the Exposure Notification API - Immuni. Different apps may have different implementations but their underlying mechanism is the same. Immuni is quite mature and was among the first Exposure Notification apps. It was downloaded more than 1.5 million times from the Google Play Store. We also explore an alternative app to Immuni called StopCovid [31] that does not utilize the Exposure Notification API. Instead, it sends BLE advertisements containing identifiers generated from the ROBERT protocol [32]. Finally, we select Find My as a specific app for iOS devices.

We setup the aforementioned devices in three different scenarios to understand the impact of various factors:

a) Scenario A: This is the most controlled one in our evaluations, yet emulates real-world BTC and BLE usage. The scenario comprises of a residential area that exhibits low to average amounts of traffic from both BTC and BLE devices. We place our test devices at random locations of this area. They stream music over BTC and transmit BLE advertisements using the apps under test. We found an additional 20 BLE advertisers and 2 BTC devices during our measurement. This low noise scenario is useful because it allows us to evaluate the performance across different devices and apps while avoiding the randomness associated with excessive RF interference and packet collisions.

b) Scenario B: This consists of an office environment with many devices operating in the 2.4 GHz spectrum, using both Wi-Fi and BT. In addition to the traffic generated in typical offices, this specific area includes two IoT labs resulting in a crowded scenario. All devices considered, this scenario presents more than 100 BLE devices advertising concurrently and transmissions from 20 different BTC devices, leading to significant interference. We placed our test devices in random locations of this office with each device using similar levels of BTC and BLE transmissions as the first scenario.

c) Scenario C: Due to limited physical space in the previous scenarios, we include another scenario in a football field about 100 meters in length. We intentionally place our devices far away at 20m, 40m, 60m in this field to measure the viability of performing the attack from a distance.

As all three scenarios described are public and not isolated, we take precautions to ensure that the information we record does not contain any private information of individuals. To that end, we only record timestamps, BLE random addresses and Service UUID to identify the application being used.

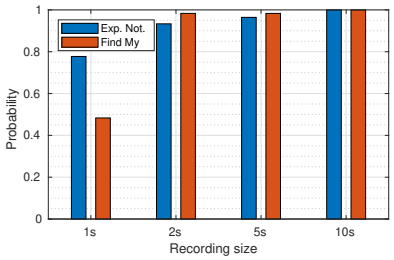


Fig. 7: Probability of linking BLE Exposure Notifications and Find My to de-anonymized BTC for different recording sizes.

VII. Attack Experimental Evaluation Results

We describe the results of the attack evaluation in the considered scenarios, exploring the impact of metrics relevant for a real world attack, such as recording time, chipset, BLE and BTC traffic or apps used.

A. Impact of Recording Time

A key attack metric, especially for Exposure Notification Service due to its low transmit power, is the duration the attacker needs to be in the vicinity of a victim. To understand its impact, we consider recording intervals of 1s, 2s, 5s and 10s. This emulates the scenario where the victim just passes by the attacker or the attacker follows the victim closely for a short duration. We use the setup described in Scenario A, and place 5 different devices each one connected to a BT device. The devices were configured to stream audio over BTC and transmit BLE advertisements. We setup the Android devices to transmit Exposure Notification advertisements and iPhone to transmit Find My advertisements. We took 10s recordings every minute for half an hour, resulting in 300s of total RF spectrum recordings. Then, we split the recordings based on the chosen interval and analyzed them independently. We computed, for each chunk, if we are able to link BTC transmissions and BLE advertisements for the target devices.

Figure 7 shows the probability of linking Exposure Notification and Find My BLE advertisements with same-device BTC transmissions. Note that the advertising period for Find My is 2s, hence the low linkability probability for 1s recordings. Nonetheless, due to iPhone X transmitting at high power, the system already achieves 95% linkability probability with 2s recordings, increased to 100% for 10s recordings. For exposure notification, we see that, even for 1s recordings, our system is able to link correctly devices advertising Exposure Notification 77% of the time. For 10s recordings, the system correctly links devices almost 100% of the time. One factor resulting in increased attack performance is the frequency of advertisements. Exposure notification advertisements are transmitted once every 260 to 280ms, and Find My every 2s. This leads to only 3-4 packets recorded in 1s and 1 packet every 2s respectively. As such, we lower the threshold for the minimum number of packets required for inference leading to false positives. The problem disappears with 10s as sufficient count of BLE advertisements are recorded.

	SGS6(1)	SGS6(2)	Pixel 3A	Xiaomi Mi8	Moto G5P	iPhoneX
1 s	0.8615	0.8659	0.9011	0.8399	0.9219	0.965
2 s	0.9358	0.9231	0.9658	0.9391	0.9524	0.9708
5 s	0.9825	1.0000	0.9831	0.9655	0.9643	0.975
10 s	1.0000	1.0000	1.0000	1.0000	1.0000	1.000
# Pkts	7627	5096	15461	3711	21539	13679

TABLE I: Probability of linking an advertisements to a BTC device for different devices and recording sizes.

B. Impact of Bluetooth Chipset

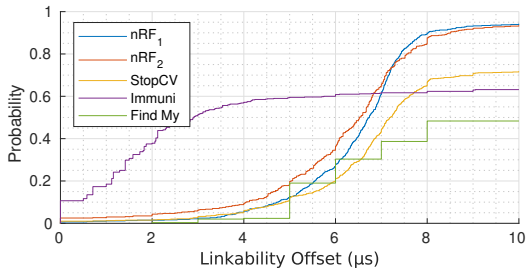
We analyze whether the attack affects all devices equally, or if certain devices are more vulnerable. We use the setup from Scenario A, with all our devices streaming audio over BTC and concurrently broadcasting BLE transmissions. For this experiment, to remove any app specific artifacts, we used nRF Connect for the BLE advertisements on Android devices. We chose nRF Connect for Android because our analysis showed it is more stable than other apps. This could be due to the fact that all other apps perform intensive cryptographic operations. Similar to the previous experiment, we recorded 10 seconds of RF spectrum every minute for 30 minutes, and then split the data into different chunks based on recording intervals.

Table I summarizes the results showing the probability of linking BLE advertisements to same-device BTC transmissions, for each test device. Note that for Apple iPhone X we use Immuni as we found it to be reliable on iOS, unlike nRF Connect which is not as configurable as its Android counterpart. The probability values range between 84% (Xiaomi Mi8) for 1s recordings to 100% for 10s. The probability for 10s is 100% for all devices in our set showing that the attack is robust across devices and chipsets. Comparing all devices, we see slight differences in linkability that we mostly attribute to the spectrum being congested.

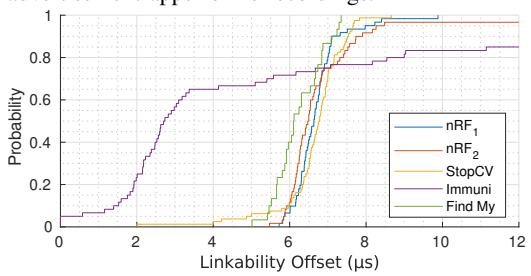
Although our results indicate that the performance of the attack is not related to the device, we did observe some artifacts during our experiments. We observed that the alignment between BLE and BTC is tighter for some devices/chipsets. For example, we find that both Samsung Galaxy S6 phones (SGS6(1) and SGS6(2)) have a very similar mismatch per packet, between 5μs to 10μs. On the other hand, Xiaomi Mi8 and Pixel 3A BTC-BLE packets are generally more tightly synchronized, with errors per packet of at most 2μs.

C. Impact of App and Advertisement Configuration

We analyze the potential impact of different advertisement configurations like transmit power and latency on attack performance. For this evaluation, we ran measurements placing a Samsung Galaxy S6 and an iPhone X in our Scenario A. To compare both the impact of different apps using BLE and the advertisement configuration itself, we ran Find My on iPhone X and Immuni, StopCovid, and two parallel BLE advertisers using nRF connect on the Samsung Galaxy S6. The first nRF Connect advertiser was setup with the same low latency configuration as StopCovid (i.e., 100ms), whereas the second advertiser was setup to emulate exposure notification



(a) Time alignment between BTC and different BLE advertisement apps for 1s recordings.



(b) Time alignment between BTC and different BLE advertisement apps for 10s recordings.

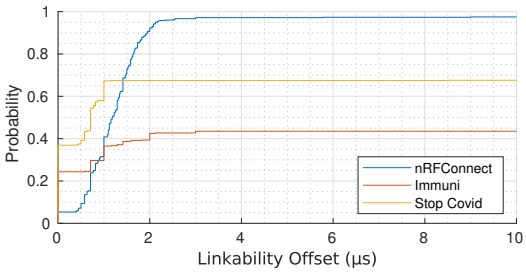
Fig. 8: Time alignment between BTC and different BLE advertisement apps for 1s and 10s recording time.

configuration latency (i.e., 250ms). Both nRF advertisers were set to transmit at higher transmission power than StopCovid and Exposure Notification. StopCovid and Find My also transmit with higher power than Exposure Notification. We place the test device within 10 meters of the attack system and recorded for 10s every minute for 1 hour. We computed the RMSE of the time alignment for each advertiser separately.

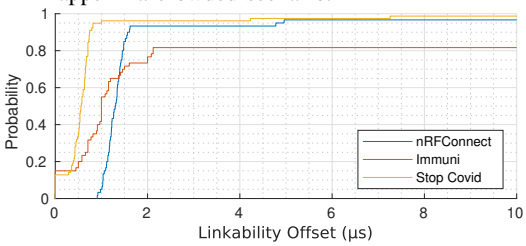
The results of this experiment are shown in Figure 8a and Figure 8b. The figures show the empirical CDF of the RMSE for the 4 different advertisements for 1s and 10s chunks, respectively. We observe that nRF Connect is more successfully linked to BTC transmissions for 1s recordings, even with low latency configurations. This is due to their higher power. For 1s recordings, StopCovid is linked 71% of the times, whereas Immuni is linked 63% of the times. Find My is able to be linked 46% of the times due to its high transmit latency (period) of 2s. For 10s recordings, we find that the performance of the attack for Immuni, StopCovid and Find My increases to 83% and ≈100% respectively. In case of Immuni, we observed that Immuni frequently stopped transmitting on channel 37 and only 10% of the 1s recordings had Immuni transmissions on channel 37. Moreover, in 15% of the recordings the decoder did not find any packets transmitted for Immuni on either channel 37 or 38, which explains the lower performance.

D. Impact of Device Density

Crowded scenarios can impose a challenge for an attacker trying to reliably perform the attack for two main reasons. First, the RF spectrum is more congested and some transmissions might not be scheduled correctly or get overshadowed



(a) Attack performance for 1s recordings for different BLE apps in a crowded scenario.



(b) Attack performance for 10s recordings for different BLE apps in a crowded scenario.

Fig. 9: ECDF of the RMSE of the linkability between BTC and different BLE applications in a crowded scenario.

by other devices transmitting at higher power. Secondly, as the number of devices grow, the probability of having another device with similar clock values increases.

We first study these issues by using the setup described in Scenario B, a noisy office environment with more than 100 BLE devices and 20 BT devices due to the proximity to two IoT labs. In order to test differences in apps and configuration parameters, we set two of our devices, Xiaomi Mi 8 and Motorola Moto G5 Plus to stream audio and run two BLE advertisers - the former with nRF Connect and StopCovid, and the latter with nRF Connect and Exposure Notification. We place these devices within 10 meters of the attack system and record 60 samples of 10s each.

Our results showcase considerable loss of BLE advertisement packets due to the congestion. For the same time duration, we were able to record 7811 advertisements for StopCovid, 3746 for Exposure Notification and 8156 for nRF Connect in the low traffic Scenario A. In Scenario B, the recorded advertisements were reduced to 2019, 497 and 5894, respectively. Consequently, the results for RMSE of the time alignment for 1s and 10s recordings in Figure 9 shows a drop in the performance of the attack. For 1s recordings (Figure 9a), the probability drops to 55% and 44% for StopCovid and Exposure Notification, respectively. For 10s recordings (Figure 9b), the attack remains robust at 90% and 82% for StopCovid and Exposure Notification, respectively. Another side effect is that, due to the number of advertisements getting reduced, the system adapts the minimum count of packets used to perform the linkability to ensure that it does not decrease the attack performance. This fact along with the increase in volume of advertisements leads to the appearance of multiple false positives. Analysing the two test devices, out of 120 measurements of 10s each, the algorithm incorrectly linked 11 advertisements.

In order to evaluate the impact of a high number of BTC devices, we take the same devices, Xiaomi Mi8 and Moto G5 Plus, to Scenario A and we isolate them from the high BLE traffic while we increase the number of BTC devices by setting up 20 additional IOGEAR Bluetooth 4.0 dongles constantly transmitting.This leads to 30 total number of BTC devices, similar to the BTC density in Scenario B, while BLE advertisements density remain low to average. Our results show that, for this scenario, Exposure Notification and StopCovid linkability probabilities increase to 76% and 71% for 1s recordings, whereas 10s recordings increase linkability above 97% for both apps. Comparing these results to the ones on Scenario B, we find that BLE is generally the bottleneck for the attack, due to its limited channels and lower power transmissions, whereas BTC has higher power and transmissions are not as impacted by interference and collisions.

E. Impact of Mobile Device BTC Traffic

Even when an active BTC session is not taking place, in many cases, a paired device may still remain connected. In such cases, the device may not transmit several BTC packets but still periodically send BTC keep-alive packets to the paired device. An attacker can exploit these keep-alive packets to execute the attack.

To test the impact of these periodic keep-alive transmissions, we used three smartphones - Xiaomi Mi8, Samsung Galaxy S6 and Motorola Moto G5 Plus in our Scenario A setup. We paired each one of these devices to a Bluetooth headset, but did not stream any music to them. As a result, these BTC connections were idle during the experiment with the exception of the keep-alive packets. The devices were all setup to transmit BLE advertisements during the entire experiment spanning 600s.

Our results indicate that devices in BTC idle mode transmit only few packets per second. For our test devices, this frequency was between 1.5 and 5 packets per second on average similar to the BLE advertisement rate. For reference, the same devices transmit about ten times more packets per second when they are streaming audio. Even with the reduced rate, our results show that the linkability per device drops, however, it is still quite good. For 10s recordings, we achieve $\approx$90% linkability for both nRF Connect and Exposure Notification advertisements. For 1s recordings, we achieve an attack performance of 80%. This suggests that the attack is robust even when the transmissions are sporadic, as long as the BTC and BLE packet timestamps are near each other.

F. Impact of Mobility and Range

As BTC transmissions can travel distances of more than 100 meters, they can be decoded from a distance without dedicated hardware. However, BLE advertisements are frequently configured with low transmit power to reduce the transmission range. A good example of this is the Exposure Notification service that has a maximum range of 10 meters. As such,

distance impacts the attack performance mainly because of the difficulty to decode BLE transmissions. As for mobility, the Doppler effect due to relative speeds between receiver and transmitter can shift the frequency of the received signal, potentially jeopardizing decoding for both BLE and BTC transmissions.

To study the viability of the attack from a distance, we setup a Xiaomi Mi8, Motorola Moto G5 Plus and iPhone X at different distances (20, 40, and 60m) from the attacker in Scenario C. The Android devices are setup to transmit high power advertisements using nRF Connect, concurrently with Immuni and StopCovid advertisements,whereas iPhone X is set up to transmit Immuni and Find My advertisements. Instead of using an omnidirectional antenna, this time we use a directional antenna with higher gain to get better coverage.

Our results show that nRF Connect advertisements are easily spotted due to high transmit power and the linkability attack has a probability above 90% even for a distance of 60m. On the other hand, we did not see any Android Exposure Notification advertisements at 60m. At 20 and 40m, the system was able to record 17 advertisements in a 300s period which is not enough recordings for a successful attack. However, for Exposure Notifications coming from the iPhone X, we are able to link them correctly with a probability of 80% even for a distance of 60m. This is in line with the results for Find My, which we are able to link with a 75% probability at 60 meters. The good results for iPhone X are due to Apple devices transmitting BLE advertisements at high power. Lastly, StopCovid, with a transmit power between the other apps, did not see many advertisements for 60m but reported a linkability of 70% and 90% at 40 and 20m, respectively.

To study the impact of mobility, we carry out our measurements in Scenario C. We set the receiver at a fixed position and we use the same setup in terms of devices and apps as we used for the distance measurements. The devices are moving within a maximum distance of 20 meters from the receiver at two different speed levels, walking and running.

Results for this experiment are summarized in Table II. We only showcase results for 5s and 10s for Find My due to its advertisement periodicity being 2s. At a walking speed, we were able to link the advertisements for all apps -except Find My- with at least 75% probability for 1s recordings. At a running pace on the other hand, linkability percentage was notably lower for certain apps. For 1s recordings, nRF Connect had still a linkability above 90% but Stop Covid dropped to 55% and the decoder missed multiple Exposure Notification packets coming from the Android device, leading to a drop in performance of the attack to 21%. Despite its lower performance on Android, Immuni on IphoneX demonstrated to be more reliable due to the high power used for its transmissions, and the attack performance was not particularly affected by speed. Similarly, Find My was not affected by dynamism, leading to high linkability values for both scenarios. For 10s recordings linkability on most apps perform well, included Exposure Notification for Android, able to be linked 77% of the time.

	Walking					Running				
	nRF	**Stop Covid**	**Imm. (An.)**	**Imm. (iOS)**	**Find My**	**nRF**	**Stop Covid**	**Imm. (An.)**	**Imm. (iOS)**	**Find My**
1s	0.95	0.75	0.80	0.90	-	0.80	0.53	0.21	0.74	-
5s	0.98	0.87	1.00	0.93	0.87	0.95	0.81	0.60	0.98	0.76
10s	1.00	0.98	1.00	0.97	0.93	0.97	1.00	0.77	1.00	1.00

TABLE II: Linkability in walking and running scenarios.

G. Summary of Attack Performance and Limitations

Evaluating the performance of the attack under varying scenarios and setups, we observe that the attack proves to be highly reliable and robust across device models and chipsets. This is true even in scenarios that would, intuitively, be challenging. For instance, the probability of linking BLE advertisements with same-device BTC transmissions is between 80% and 95% in a crowded environment with more than 100 BLE advertisers and significant congestion. Another challenging scenario is when a BTC connection on a device is idle and just transmitting sporadic keep-alive messages. In this scenario, an attacker is able to correctly link BLE advertisements 90% of the time.

The results indicate that BLE advertisements transmitted with higher power have higher probability of linkage with BTC transmissions. For example, the Find My app uses high transmit power and high transmission period (2s) and is more vulnerable to the attack than apps using the Exposure Notification service on Android. This is because, although the Exposure Notification service transmits more frequently, every 250 - 280ms, the transmit power is low, increasing the chances of the decoder missing packets.

To make matters worse for victim(s), the attacker does not need to obtain sufficient data to execute the attack. Our results indicate that the probability of linking correctly a device is already above 90% for 10s recordings in most scenarios. Shorter recording times do see a performance drop, but the attack is still successful even for 1s recording times. This is especially true in low traffic scenarios, where 1s recordings have a linkability probability of above 75%.

In terms of limitations, we note that, without improvements to the BLE/BTC decoders, it becomes difficult to exploit the attack at distances above 80 meters and as speed increases. Scaling the exploitation of the vulnerability would require additional steps. For instance, the adversary could conduct the attacks at the entrance of a testing center or crowded spaces. Alternatively, we believe the attack can be implemented as a mobile (without SDR) within the Nexmon framework [27]. Finally, some protocols [9] support enhancements based on a k-out-of-n secret sharing. We expect such schemes to limit the exploitation of the vulnerability to users with low mobility. On the other hand, since an adversary only needs k out of n advertisements, it might also extend the range of the attack if the shares results in shorter messages than the original advertisements.

Considering the impact of the vulnerability on privacy, the attacker can link BLE advertisements to the device BDADDR. While the impact on Find My might be limited, an attacker

can infer the BDADDR of infected users devices. A path to full de-anonymization is discussed in Section IV-E

VIII. Countermeasures

We discuss countermeasures, their design, and implementation in Android to reduce the impact of the vulnerability.

A. Measures for Reducing Attack Performance

The identified vulnerability exploits fundamental design decisions in the Bluetooth standard. As such, complete mitigation of the attack may require significant changes, unlikely to happen in the near future. Below, we outline some countermeasures that can be implemented for the Android OS to reduce the attack performance.

1) Disable BTC when inactive: The Android OS does not automatically disable BTC on the device when inactive. This increases the likelihood of the BTC-BLE linkage attack as the device can connect to any paired device in proximity. Note than any app that requests the `android.permission.BLUETOOTH_ADMIN` permission can also enable BTC on the device. Therefore, the functionality to proactively disable BTC should be implemented directly into the Android OS.

2) Reduce BLE Tx power when BTC connected: The Android OS currently does not restrict concurrent BTC and BLE transmissions. When a smartphone user is using BTC (e.g., streaming music), disabling BTC will result in usability concerns. Disabling BLE is also not feasible as that may disable health applications (e.g., Exposure Notification, Samsung Health) that use BLE for their services. One potential measure to limit the attack radius (e.g., to 6-10 meters) can be to reduce the BLE Tx power when a BTC connection is active and restore original Tx power once BTC is disabled.

3) Restart Advertisements instead of Randomization: One measure to protect against BLE pre- and post- randomization linkage attacks can be to restart the advertisements every few minutes instead of randomizing the MAC address. By introducing a small random delay before each restart, the advertisements can be made to switch to a different clock and change the timing information.

4) Minimize BLE Tx power before Randomization: Recall that the clocks used for channel hopping drift over time. As such, reducing the BLE advertisement Tx power to a minimum for a moment before randomization and restoring power after randomization can force a distant attacker to lose BLE advertisements and synchronization.

B. Implementing an Android System for Attack Mitigation

Our objective with the Android mitigation system was to ensure that the system (1) does not affect the functionality of existing apps that use BTC and BLE, and (2) minimizes changes required to the Android Bluetooth stack. In terms of measures, we focused on the aforementioned measures like controlling the BTC state based on connection status, reducing the BLE Tx power when BTC is active, and minimizing the BLE Tx power before MAC randomization. Note that the system does not focus on restarting BLE advertisements (see section VIII-A3) as this measure violates our objective of not affecting app functionality. We also note that certain mitigations may not work against a determined attacker who uses powerful antennas to execute the attack from a distance. Our implemented system addresses the above measures utilizing two levels of abstraction. First, we extend the current Android Bluetooth stack to address problems that allows an attacker to launch the attacks more easily. Next, we build upon these extensions to opportunistically change the state of BTC and BLE transmissions based on how the medium is being utilized by the user and installed apps. The change in state also generates user notifications that users can view on their devices. These notifications show (and warn) the users when BTC and BLE are concurrently used on the system.

1) Extensions to the Android Bluetooth stack: The Android Bluetooth stack presents a serious problem that both BTC and BLE are controlled using the same API. There are only two possible states - BTC Off, BLE Off (`BTC_OFF_BLE_OFF`) reachable by invoking the `disable` API provided by the Android SDK `BluetoothAdapter` implementation; and BTC On, BLE On (`BTC_ON_BLE_ON`) by invoking the `enable` API provided by the same class. There are currently no public APIs that apps can invoke to control BLE directly when BTC is disabled. This forces installed apps to enable BTC on the device even when they do not require any BTC services.

We solve the above problem in our system by decoupling BTC and BLE, enabling BLE on the device even when BTC is disabled. More precisely, our system introduces a new state BTC Off, BLE On (`BTC_OFF_BLE_ON`) that allows installed apps to use BLE without reliance on BTC. We implemented this new state by creating public wrappers around hidden APIs in the Android source code using Java Reflection. The Android Bluetooth stack implements methods (e.g., `enableBLE`, `disableBLE`, `isLeEnabled`) in the `BluetoothAdapter` class that can be invoked to enable and disable BLE on the device. These methods have a scope defined as `public` in the source code, it is just that the APIs are not documented in the Android SDK. We must also note that the above methods are protected using the same permissions as BTC (`android.permission.BLUETOOTH_ADMIN`). As such, our new state does not introduce any security vulnerabilities in the Android ecosystem. Implementing this wrapper does not require any changes to the Android OS, nor does it require rooting the device or installing third-party frameworks. We will make this wrapper open-source as an Android library to prevent apps from invoking the `BTC_ON_BLE_ON` state (and enabling the attack) in the future.

2) Implementing Measures limiting the attacks: The states `BTC_OFF_BLE_OFF`, `BTC_OFF_BLE_ON`, and `BTC_ON_BLE_ON` can be utilized to implement the protections from the attack. To implement the countermeasures, we require the capability of updating an app's BLE transmissions without instrumenting the app or changing its functionality. In all versions of Android, this is infeasible due to obvious security reasons. To achieve our objective, we implemented the system

as a module for the Xposed framework [33]. The framework enables our system to insert code into Android SDK APIs before or after the API invocation, the BLE Advertiser API in our case. We preferred this approach over modifying the Android source to quickly ship this protection to millions of users who already use Xposed on their devices.

Our system tracks the count of BTC connections, BLE connections and BLE advertisers to determine the current state of the device. The Android OS implements *broadcast intents* that all apps can register to receive certain device updates. Our system uses the intent action `BOOT_COMPLETED` to track when a device has booted up and enters the default state of `BTC_OFF_BLE_ON`. To track the count of BTC connections, the system tracks the intent actions `ACL_CONNECTED` and `ACL_DISCONNECTED` which are broadcast whenever a BTC paired device connects and disconnects with the device. To track BLE connections, the system tracks the intent actions `BLE_ACL_CONNECTED` and `BLE_ACL_DISCONNECTED`. The above BLE intents are not documented in the Android SDK and we found these intents during our analysis of the Android code. Unfortunately, there are no intents to track BLE advertisers. As such, we rely on the Xposed framework to insert counters into the methods `startAdvertisingSet` and `stopAdvertisingSet` of the `BluetoothLeAdvertiser` class to track the count of BLE advertisers. These methods are also used to record an app's advertising parameters (e.g., service UUID, `Tx` power, service data) in order to modulate the BLE `Tx` power.

The system implements state transitions that are triggered whenever the counts are updated, i.e., whenever any BTC and BLE connections or disconnections occur. In many cases, even when the counts change, the transitions do not occur as the device state remains the same (e.g., two apps now broadcasting BLE advertisements instead of one). At each update, a notification is generated that the user can view on the device to see a list of all BTC, BLE connections and BLE advertisements. Specifically, the system implements the following two important transitions and parameter updates -

- `BTC_OFF_BLE_ON` → `BTC_ON_BLE_ON`: This transition occurs when the user or an app enables BTC on the device. If no BTC connection is established within a reasonable time, the system switches back to `BTC_OFF_BLE_ON` state. To reduce the BTC-BLE linkage attack radius, all BLE advertisers active on the device switch to a low power transmission mode. In this mode, the advertisements just have enough power to be reachable within a 6-10 meters radius of the device. The `Tx` power update is achieved using the Xposed framework by inserting custom code in the `setTxPowerLevel` method of the `AdvertisingSetParameters` class of the Android SDK.
- `BTC_ON_BLE_ON` → `BTC_OFF_BLE_ON`: This transition occurs when all paired BTC devices have disconnected from this device. As BTC is disabled, the possibility of BTC-BLE linkage attacks is eliminated. Therefore, all BLE advertisers are switched back to their regular transmission powers. As the risk of pre- and post- randomization linkage attack still exists, the system periodically switches all advertisers to minimum power transmission mode for a small duration. This is done so that an attacker outside this minimum range loses synchronization with the device and is unable to link the pre- and post- randomized MAC addresses.

The system is intended to reduce the attack performance rather than eliminating the attack. The system does not enforce changes to apps that currently enable BTC to transmit BLE advertisements. Instead, we intend to spread awareness of this problem and provide developers a library to easily switch to the more secure `BTC_OFF_BLE_ON` state. We validated the library by implementing a BLE Advertiser app without using any BTC code and observed that it works reliably[1]. We believe that eliminating this attack would require significant changes to the BTC and BLE chipsets. Such extensive changes are outside the scope of this work.

C. Impact of the Mitigation System on Attack Performance

To verify the mitigation system, we loaded it on a Google Pixel smartphone. We performed the same experiment twice for 200 seconds each - once using a factory image and another one using our mitigation system. In both cases, we paired the smartphone with a headset and streamed audio to it with pauses in between. The device was setup to transmit BLE advertisements at high transmit power. The device was placed at a 15 meter distance from the attack system.

In the first experiment using the factory image, the attack system was able to decode 826 advertisements transmitted from the device. In addition, 1497 BTC transmissions were also recorded. In the second experiment using our mitigation system, the attack system was not able to decode any BLE advertisements. This was because the mitigation system automatically switched the BLE advertiser to a low power mode to mitigate the BTC-BLE linkage attacks. There was no degradation in BTC audio streaming, as the attack system still recorded 2035 BTC packets during the experiment. We verified the range of the advertisements using a BLE scanner app and found that an attacker would need to be as close as 6 meters to the victim(s) to execute a successful attack. The BLE advertiser restored the transmit power automatically when the BTC headset was disconnected.

We also modified the StopCovid app source code to determine how much effort app developers would require to use our more secure `BTC_OFF_BLE_ON` state. We note that, with the exception of method name changes, we were able to run StopCovid without BTC by modifying only 6-8 lines of code. As such, we believe that app developers can easily protect their users from this attack by using our mitigation system.

IX. Conclusion

We reveal that apps that rely on BLE advertisements are vulnerable to tracking by linking the advertisements to Bluetooth Classic frames, and ultimately to the device's globally

[1]A video demonstration is available at https://www.dropbox.com/s/4k1lquq2m8rzzua/BTController.mp4?dl=0.

unique identifier (BDADDR). Despite challenges in exploiting the coupling between BTC and BLE, our analysis allowed us to develop a proof-of-concept end-to-end attack that is highly reliable across devices, mobile apps, type of traffic, density of devices and range. We proposed, implemented, and evaluated a set of effective countermeasure for Android, and disclosed our findings and mitigations to Apple and Google.

REFERENCES

[1] K. Fawaz and K. G. Shin, "Location privacy protection for smartphone users," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014.

[2] Senate Judiciary Committee, "S.2171 - Location Privacy Protection Act of 2014," https://www.congress.gov/bill/113th-congress/senate-bill/2171, 2014.

[3] 3rd Generation Partnership Project (3GPP), "Technical Specification Group Services and System Aspects: System Architecture for the 5G System; Release 16," 2020.

[4] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. Rye, and D. Brown, "A study of mac address randomization in mobile devices and when it fails," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, pp. 365 – 383, 2017.

[5] Core Specification Working Group, "Bluetooth Core Specification: Revision 5.2," 2019.

[6] M. Cominelli, F. Gringoli, P. Patras, M. Lind, and G. Noubir, "Even black cats cannot stay hidden in the dark: Full-band de-anonymization of bluetooth classic devices," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 534–548.

[7] L. Ferretti, C. Wymant, M. Kendall, L. Zhao, A. Nurtay, L. Abeler-Dörner, M. Parker, D. Bonsall, and C. Fraser, "Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing," *Science*, vol. 368, no. 6491, 2020. [Online]. Available: https://science.sciencemag.org/content/368/6491/eabb6936

[8] Apple Inc. & Google LLC, "Privacy-Preserving Contact Tracing - Apple and Google," https://www.apple.com/covid19/contacttracing/, 2020.

[9] C. Troncoso, M. Payer, J.-P. Hubaux, M. Salathé, J. Larus, E. Bugnion, W. Lueks, T. Stadler, A. Pyrgelis, D. Antonioli, L. Barman, S. Chatel, K. Paterson, S. Čapkun, D. Basin, J. Beutel, D. Jackson, M. Roeschlin, P. Leu, B. Preneel, N. Smart, A. Abidin, S. Gürses, M. Veale, C. Cremers, M. Backes, N. O. Tippenhauer, R. Binns, C. Cattuto, A. Barrat, D. Fiore, M. Barbosa, R. Oliveira, and J. Pereira, "Decentralized privacy-preserving proximity tracing," arXiv:2005.12273 [cs.CR], 2020.

[10] R. L. Rivest, D. Weitzner, L. Ivers, I. Soibelman, and M. Zissman, "Pact: Private automated contact tracing," 2020.

[11] R. L. Rivest, J. Callas, R. Canetti, K. Esvelt, D. K. Gillmor, Y. T. Kalai, A. Lysyanskaya, A. Norige, R. Raskar, A. Shamir *et al.*, "The pact protocol specification," *Private Automated Contact Tracing Team, MIT, Cambridge, MA, USA, Tech. Rep. 0.1*, 2020.

[12] J. Chan, D. Foster, S. Gollakota, E. Horvitz, J. Jaeger, S. Kakade, T. Kohno, J. Langford, J. Larson, P. Sharma, S. Singanamalla, J. Sunshine, and S. Tessaro, "Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing," arXiv:2004.03544 [cs.CR], 2020.

[13] R. Raskar, I. Schunemann, R. Barbar, K. Vilcans, J. Gray, P. Vepakomma, S. Kapa, A. Nuzzo, R. Gupta, A. Berke, D. Greenwood, C. Keegan, S. Kanaparti, R. Beaudry, D. Stansbury, B. B. Arcila, R. Kanaparti, V. Pamplona, F. M. Benedetti, A. Clough, R. Das, K. Jain, K. Louisy, G. Nadeau, V. Pamplona, S. Penrod, Y. Rajaee, A. Singh, G. Storm, and J. Werner, "Apps gone rogue: Maintaining personal privacy in an epidemic," arXiv:2003.08567 [cs.CR], 2020.

[14] S. Vaudenay, "Centralized or decentralized? the contact tracing dilemma," Cryptology ePrint Archive, Report 2020/531, 2020, https://eprint.iacr.org/2020/531.

[15] H. Cho, D. Ippolito, and Y. W. Yu, "Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs," arXiv:2003.11511 [cs.CR], 2020.

[16] N. Trieu, K. Shehata, P. Saxena, R. Shokri, and D. Song, "Epione: Lightweight contact tracing with strong privacy," arXiv:2004.13293 [cs.CR], 2020.

[18] R. Sun, W. Wang, M. Xue, G. Tyson, S. Camtepe, and D. C. Ranasinghe, "An empirical assessment of global covid-19 contact tracing applications," arXiv:2006.10933 [cs.CR], 2020.

[19] J. Bell, D. Butler, C. Hicks, and J. Crowcroft, "Tracesecure: Towards privacy preserving contact tracing," arXiv:2004.04059 [cs.CR], 2020.

[20] Presidenza del Consiglio dei Ministri, "Immuni - Commissario straordinario per l'emergenza Covid-19," 2020. [Online]. Available: https://github.com/immuni-app

[21] Federal Office of Public Health FOPH, "SwissCovid app," 2020. [Online]. Available: https://foph-coronavirus.ch/swisscovid-app/

[22] Government of Canada, "COVID Alert, Canada," 2020. [Online]. Available: https://www.canada.ca/en/public-health/services/diseases/coronavirus-disease-covid-19/covid-alert.html

[23] California Dept of Technology, "California COVID Notify," 2020. [Online]. Available: https://play.google.com/store/apps/details?id=gov.ca.covid19.exposurenotifications

[24] A. Greenberg, "The clever cryptography behind apple's 'find my' feature," 2019. [Online]. Available: https://www.wired.com/story/apple-find-my-cryptography-bluetooth/

[25] M. Green, "How does apple (privately) find your offline devices?" 2019. [Online]. Available: https://blog.cryptographyengineering.com/2019/06/05/how-does-apple-privately-find-your-offline-devices/

[26] Bluetooth SIG, Inc., "Bluetooth Market update," https://www.bluetooth.com/bluetooth-resources/2019-bluetooth-market-update/, 2019.

[27] M. Schulz, D. Wegemer, and M. Hollick, "Nexmon: The c-based firmware patching framework," 2017. [Online]. Available: https://nexmon.org

[28] M. Stute, S. Narain, A. Mariotto, A. Heinrich, D. Kreitschmann, G. Noubir, and M. Hollick, "A billion open interfaces for eve and mallory: Mitm, dos, and tracking attacks on ios and macos through apple wireless direct link," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019.

[29] G. Celosia and M. Cunche, "Discontinued privacy: Personal data leaks in apple bluetooth-low-energy continuity protocols," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, 2019.

[30] "Bluetooth Low Energy SDR C++ Library," 2020. [Online]. Available: https://github.com/jocover/BLESDR

[31] INRIA, "Stop Covid, France," 2020. [Online]. Available: https://gitlab.inria.fr/stopcovid19

[32] Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, et al., "ROBERT: ROBust and privacy-presERving proximity Tracing." 2020. [Online]. Available: https://hal.inria.fr/hal-02611265/document

[33] Xposed Framework, "The Xposed Framework Source Code," https://github.com/rovo89/XposedInstaller, 2017.