Deanonymizing Device Identities via Side-channel Attacks In Exclusive-use IoTs: A Reality Today, A Challenge Tomorrow

Christopher Ellis, Yue Zhang, Mohit Jangid, Zhiqiang Lin
The Ohio State University

***Abstract*—Wireless technologies such as Bluetooth and Wi-Fi are crucial to the Internet of Things (IoT) as they enable devices to communicate without physical connections. However, this wireless convenience also exposes data exchanges to potential observation by attackers, leading to security and privacy threats such as device tracking. To mitigate these threats, protocol designers often employ strategies like address and identity randomization. While previously considered effective against tracking attacks, this paper demonstrates that these attacks remain a significant threat due to a historically overlooked flaw in exclusive-use wireless communication. Specifically, we define exclusive-use as a scenario where devices are designed to provide functionality only to an associated or paired device. The unique communication patterns in these relationships create a single-bit side-channel observable by attackers, revealing whether two devices "trust" each other. This information leak deanonymizes devices, enabling tracking despite modern countermeasures. We introduce our tracking attacks as IDBLEED and demonstrate that BLE and Wi-Fi protocols, which support confidentiality, integrity, and authentication, are vulnerable to deanonymization due to this fundamental flaw in exclusive-use wireless communication. Finally, we propose a generalized mitigation framework through the addition of an anonymization layer and quantify its performance.**

1. Introduction

Wireless technologies such as Bluetooth and Wi-Fi play a crucial role in the Internet of Things (IoT), with various applications for smart homes, entertainment, health care, retail, and personal fitness. However, the convenience of wireless communication also makes data exchanges between IoT devices vulnerable to observation by attackers through sniffing, enabling security and privacy attacks such as device location tracking. For example, modern smartphones that use Bluetooth Low Energy (BLE) for communication [1] with accessories, such as earbuds, can have their packets observed by an attacker with a low-cost sniffer. These packets include a MAC address—the smartphone's identifier—that can later be used to track their owner's location, given the low probability of address collisions [2], [3].

To defend against tracking attacks, protocol designers introduced address or identity randomization. For example, the Bluetooth Special Interest Group (SIG) has recommended MAC address randomization, which periodically changes its MAC address (e.g., every 15 minutes [4]). This countermeasure makes it significantly more difficult to track devices through MAC address tracking attacks across time cycles, particularly in areas with heavy wireless congestion from multiple devices.

However, tracking is still possible. For instance, implementation flaws have been shown to potentially static information used for tracking [5], [6], [2]. Recently discovered *BAT* attacks [7] also show BLE tracking is possible through exploiting a specification flaw in randomized MAC address generation, observing differences in communication between devices using allowlists. Inspired by this BLE focused research, we investigated IoT wireless communication protocols from a general perspective and surprisingly uncover a historically overlooked characteristic fundamental to devices that exclusively communicate: the difference in behavior between trusted and untrusted devices at multiple steps in common protocols, such as BLE and Wi-Fi.

More specifically, we introduce the *exclusive-use* concept as the scenario in which one device exclusively provides functionality to a singular or small group of trusted devices. Exclusive-use devices establish a unique, trusted association with their peer, allowing them to recognize and securely communicate with each other while preventing unauthorized access. Because trusted associations remain even when devices change addresses, we have discovered *a single-bit side-channel leak during encryption, integrity verification, authentication, and auto-connection communication stages.* Observing the differences in traffic patterns at these stages between exclusive-use devices, regardless of protocol or randomization countermeasures, deanonymizes the device and enables user tracking—an attack we introduce as IDBLEED.

For example, consider the scenario where an attacker intends to deanonymize and track Alice's smartphone, despite it using modern address randomization countermeasures. Alice leaves home and takes her smartphone but leaves her previously paired smartspeaker. Since her smartspeaker is exclusive-use, it only recognizes connection requests from Alice's smartphone and rejects all requests from untrusted devices. The attacker deploys multiple relay devices at different locations that forward packets between the smartphones and smartspeaker. If a distant smartphone can successfully communicate with the smartspeaker through the relay, the attacker knows that the

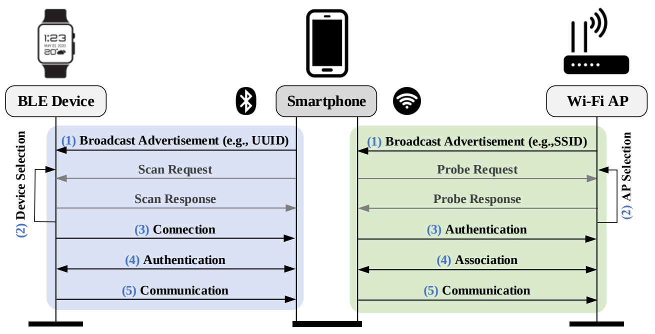

Figure 1. Typical BLE and Wi-Fi workflows.

smartphone must be Alice's phone and its location—the two devices would fail to communicate otherwise.

While IDBLEED attacks are applicable to any exclusive-use communication protocol, we evaluated three ubiquitous wireless communication technologies, BLE and Wi-Fi, used by numerous IoT devices with exclusive-use characteristics for the existence of vulnerabilities to privacy and specifically tracking attacks. For example, we identified BLE devices that use *Connection Signature Resolving Key* (*CSRK*) to verify data integrity are vulnerable. While *Long Term Key* (*LTK*) protects data confidentiality, it also exposes a single-bit side-channel due to difference in communication patterns between trusted and untrusted devices.

Devices must use a trust mechanism to provide their functionality, which inherently creates exclusive-use and allows attackers to break the anonymity of devices. A sufficient countermeasure requires protocol designers to further consider the classical dilemma of balancing security and privacy with functionality. We therefore propose a generalized mitigation that provides a framework for an Anonymization Layer (AL) that enables devices to communicate similarly with both trusted and untrusted devices, cloaking exclusive-use characteristics. This layer removes the need for sniffable source destination information by providing implicit addressing, as well as packet encryption to prevent side-channels at all phases of communication after pairing. We evaluate and quantify the mitigation overhead using an exemplar protocol simulation.

Contributions. Our contributions in this paper are threefold:

- **Novel Vulnerability (§3).** We are the first to demonstrate the vulnerability in a ubiquitous wireless communication scenario we call *exclusive-use*, in which traffic pattern differences at certain stages reveal their trusted relationship. We focus on IoT devices and show that this fundamental and overlooked flaw can be exploited by attackers through either passive observation of wireless traffic or actively relaying (or replaying) packets.
- **Concrete Attacks (§4).** We confirm through protocol and real-world packet analysis that widely used wireless technologies, BLE and Wi-Fi, are vulnerable to tracking attacks by exploiting exclusive-use characteristics to deanonymize devices, which we introduce as IDBLEED. Further, these attacks are feasible at low-cost as they exploit protocol traffic pattern vulnerabilities and do not require advanced device compromise or malware.
- **Mitigation Framework (§5).** We propose a generalized mitigation framework that introduces an Anonymization Layer that supports anonymous communication between devices over broadcast channels using ephemeral identifiers, removes the need for destination addresses, and addresses the single-bit side-channel leak through pseudo-communication with untrusted devices. We implement a simulation and evaluate its performance overhead.

2. Background

2.1. Wireless Communication Protocols

Wireless technologies are crucial to IoT as they allow devices to freely move in space without the burden of a physical wire by communicating and exchanging data over the air. While many types of wireless protocol exist, we particularly focus on Bluetooth Low Energy (BLE) [1] and Wi-Fi [8] in this paper due to their ubiquitous presence, even beyond IoT. We present background below (more details on their workflows can be found in Appendix §A).

Bluetooth Low Energy. BLE is used by devices for bidirectional wireless links, such as wireless speakers, car infotainment systems, smart home devices, and smartwatches (Figure 1). Due to its similar transmission range and relatively low power requirements compared to classic Bluetooth, BLE is also very common for IoT devices. BLE devices take on either a central or peripheral role per connection, which determines protocol responsibilities during communication.

Wi-Fi. Wireless Fidelity (Wi-Fi) is a popular family of protocols based on IEEE 802.11 standards that allows devices to wirelessly connect to networks. Wi-Fi Access Points (AP) act as servers that allow connections from clients, such as computers, smartphones, or smartdevices, using a variety of authentication mechanisms to access the network, illustrated in Figure 1. Other Wi-Fi protocols, such as Wi-Fi Direct, are similar to classic Wi-Fi but turn traditionally client devices into APs. A common use case is smartphones creating ad-hoc networks to exchange data rather than use cellular service.

2.2. Identity and Address Randomization

Networks rely on unique device identities to route unicast data to its intended destination. The most common type of network device identifier is a Media Access Control (MAC) address. Much like a recipient's name on a mail envelope, MAC addresses are included in a transmitted packet.

Unfortunately, MAC addresses are inherently vulnerable to sniffing when packets are sent over the air using a wireless technology. This potentially allows attackers to identify and track devices, raising privacy concerns for users. For example, an attacker can sniff packets exchanged between

Address Type	Static	Rotation Cycles	Vulnerable To Tracking	Example Devices
PA	✓	∞	✓	Most IoT devices (e.g.,Smart locks)
SRA	✗	Device-Specifc	✗	Office supplies (e.g., Keyboards)
RRPA	✗	1 - 15 mins	✗	Smartphones (e.g., Android and iOS)
NRPA	✗	1 - 15 mins	✗	Bluetooth Beacons

TABLE 1. SUMMARY OF FOUR TYPES OF BLE ADDRESSES

a paired smartwatch and smartphone that may use unique MAC addresses and determine when a victim has arrived or departed a particular location.

As a countermeasure, some devices now use address randomization to support enhanced privacy and defend against tracking while still allowing devices to be addressable. Typically this involves a protocol that periodically generates new, random identifiers, while still allowing communication with their trusted devices. We now review address randomization techniques used by BLE and Wi-Fi. Which technique is used by a real-world device is determined primarily by user settings or developers providing backwards compatibility, or limited by technology.

BLE Address Randomization. BLE uses four different types of addresses, three of which are randomly generated to defend against tracking attacks, as shown in Table 1.

- *Public Address* (PA): a globally static address assigned by a manufacturer to serve as a unique device identity. The PA is vulnerable to MAC address tracking attacks since it never changes.
- *Static Random Address* (SRA): a random address generated by a device upon reboot or reset. If that never or very rarely occurs, the SRA is vulnerable to tracking.
- *Resolvable Random Private Address* (RRPA): a random address periodically generated by a device that is resolvable by paired devices using a previously exchanged *Identity Resolving Key* (IRK) in a simple challenge-response protocol.
- *Non-Resolvable Private Address* (NRPA): a random address periodically generated but, depending on the implementation, is never intended to be resolvable.

Wi-Fi Address Randomization. Wi-Fi MAC address randomization is widely used in recent versions of Android [9] and iOS operating systems [10]. Android uses one of two methods: the first generates a random MAC address for each network based on its AP beacon attributes that does not change over time. This safeguards against tracking static MAC addresses across different networks. The second generates a random MAC address for a network after 24 hours, but only if the device has disconnected, which provides a defense against an attacker sniffing traffic. iOS 14 and later also use randomly generated MAC addresses for each Wi-Fi network in a similar fashion. Additionally, Wi-Fi probe beacons for discovering nearby APs use a randomized address to prevent tracking from sniffing a device that is discovering APs.

To ensure randomly generated addresses are still recognized by APs, both Android and iOS may use other Wi-Fi authentication protocols such as Extensible Authentication Protocol (EAP). In EAP, the device sends a request to join the network, the AP sends a challenge, and the device resolves this challenge to prove its identity.

2.3. Protocol Features

Network protocols exist to provide a service and communication between devices, enhanced by features established at various stages. These include confidentiality, integrity, authenticity, and auto-connection, summarized below as a basis for our attacks.

Confidentiality. Encryption is used by various wireless protocols to provide data confidentiality. When two devices establish communication, they typically negotiate one or more secret keys to verify each other or encrypt data. For example, BLE devices may negotiate an *LTK* during the pairing process and later use it to derive a session key for data encryption. Only devices with the same *LTK* can derive the same session key and access the encrypted traffic, enforcing the trusted relationship.

Integrity. Data signature or message authentication code (MAC) verification using shared secret cryptographic keys ensure data integrity—received data is as intended by the originator. For example, BLE supports verifying unencrypted data using a *CSRK*. The key is sent from one device to the other during the pairing process and used to produce a MAC that is appended to the message and sent as a packet payload. Each device in a BLE ad-hoc network has a unique *CSRK* with every other device, ensuring exclusive-use and protecting messages from being altered by unauthorized parties.

Authenticity. Authentication is a process to verify the identity of devices and allow access to a network or resource. It typically involves a challenge-response protocol in which two devices use a previously shared and secret cryptographic key. One sends a challenge message with a MAC generated using the shared key. The other device uses the shared key to solve the challenge, responding with the message. The first device verifies the code and authenticates the other if it is valid. Authentication is often used in conjunction with encryption, but each serves different purposes.

Auto-Connection. Automatically connecting to trusted networks or devices enhances user convenience and experience. Smartphones often auto-connect to familiar Wi-Fi networks, like a home router, when within range, detected via probing beacons. Likewise, an IoT companion app on a smartphone can automatically link with its paired BLE device when nearby. This seamless connectivity, often occurring in the background or when the phone is inactive, spares users from manually selecting networks or devices and waiting to connect.

3. IDBleed Tracking Attacks

3.1. Deanonymization via Exclusive-use

We define *exclusive-use* as two or more trusted devices that are observably associated due to their communication patterns. We observe an exclusive-use device will communicate differently with an associated, trusted peer than with any other. This common protocol design protects against security threats by only allowing devices with an established trust relationship, typically owned by the same user or associated group, to access specific resources and services.

However, we observe the communication patterns of this design also inherently deanonymizes a device and enables an attacker to track victims. While numerous works exist on tracking attacks [2], [3], [5], [6], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], we are confident to be the first exploring and bringing awareness to this historically overlooked and nuanced characteristic fundamental to exclusive-use devices. We discuss key differences with related works in §7. *We further realize numerous types of protocols exhibit exclusive-use characteristics and therefore, our concept is not limited to only IoT but is ubiquitous in communication.*

Effectively, a device's exclusive-use characteristics are observed via a single-bit side-channel leaked during various communication stages. One of two possible outcomes during communication potentially deanonymizes a device: a successful path means a trusted relationship exists while a failure path means it does not. We specifically observe this leak during stages handling confidentiality, integrity, authenticity, and auto-connection.

Combining the exclusive-use side-channel observation with relay and replay techniques produces a new, modern tracking attack we introduce as IDBLEED. Attackers deanonymize and track victims despite modern countermeasures such as MAC address randomization.

3.2. Threat Model

Goal & Motivations. The attacker's goal is to track the location of a specific person or member of an associated group. Technically, the attacker deanonymizes their victim's smartphone assumed to be on their person, and thus learns of their presence at a specific location at a point in time. Their motivations may range from tracking locations or pattern-of-life analysis of family members, friends, political figures, or others in power. IDBLEED enables covertly discovering a victim's presence at specific locations digitally, without physically following them and risk raising suspicion. Examples locations include a home, office, daycare, coffee shop, airport, state, or country. An attacker can range in technical skill, from a lone hobbyist with intermediate coding skills and knowledge in wireless sniffing to an expert group of nation state actors. IDBLEED is executed using one of two methods: Passive (M1) or Active (M2).

Assumptions & Requirements. The two prerequisite assumptions and requirements for IDBLEED attacks are:

1) The victim's devices are exclusive-use and formed a trusted relationship. This implies a unique association between the user and the devices have distinct success and failure communication patterns.
2) The attacker can sniff and potentially relay and retransmit communication traffic between devices.

3.3. Attack Method Workflows

We now present the generalized Passive and Active IDBLEED method workflows using honest users *Alice*, *Bob*, *Charlie*, and attacker *Eve*. Each represent a device capable of exchanging information using a sniffable communication channel, for example, $\texttt{REQ}(A \rightarrow B)$ and $\texttt{RSP}(B \rightarrow A)$. Further, *Alice* and *Bob* are exclusive-use and trusted, thus associated. *Charlie* is simply any, and an infinite number of, non-associated devices. *Eve* sniffs, forwards, and re-transmits their communication traffic. These workflows provide a basis for our real-world analysis in §4.

(M1) Passive Deanonymization. *Bob* is exclusive-use and trusts *Alice*, and therefore only responds to their requests but ignores *Charlie*'s. Meanwhile, *Eve* is able to observe the communication between all three users. Shown in Figure 2, at t_1, *Alice* uses identity ID_{A1} to send request $\texttt{REQ}(A1 \rightarrow B)$, which *Bob* responds with $\texttt{RSP}(B \rightarrow A1)$. *Charlie* also sends a request $\texttt{REQ}(C1 \rightarrow B)$ but is ignored. At t_2, both *Alice* and *Charlie* change their identities to ID_{A2} and ID_{C2}, respectively. Once again, *Alice* and *Charlie* send requests to *Bob*. *Eve* observes the requests and responses between *Alice* and *Bob*, as well as *Bob* not responding to *Charlie*'s requests. As such, these different responses leak a single-bit side-channel that leads *Eve* to conclude ID_{A1} and ID_{A2} belong to *Alice*. At this moment, *Eve* breaks the anonymity offered by identity randomization, leaving *Alice* vulnerable to tracking.

While M1 is effective when traffic from two devices can be sniffed, it is no longer suitable once they move outside their communication protocol's transmission range. For example, one may carry their smartphone to a coffee shop while their BLE-paired smartspeaker remains at home. Further, the passive attack relies on one device maintaining a static identity, unless communication is observed between the points of asynchronous identity randomization between the two communicating devices, as explored in [7]. However, these limitations are resolved in the more powerful active IDBLEED attack.

(M2) Active Deanonymization. The active IDBLEED attack utilizes a relay network setup by *Eve* at various locations where *Alice* may be. As shown in Figure 3, consider when *Eve* knows that *Alice* exclusively communicates with *Bob*. At time t_3, *Alice* is away from *Bob*. At one relay location, an anonymized device initiates a request $\texttt{REQ}(X \rightarrow Y)$, *Eve* captures and relays the packet to *Bob*. If *Bob* responds with $\texttt{RSP}(B \rightarrow X)$, *Eve* now knows

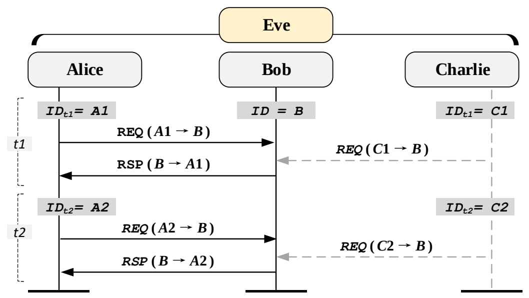

Figure 2. Passive Deanonymization

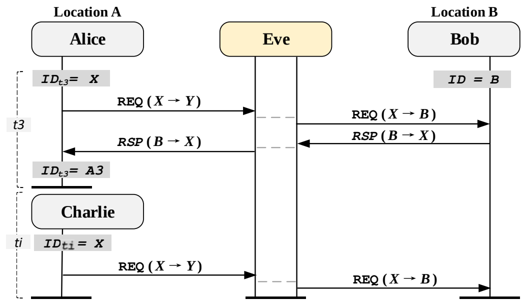

Figure 3. Active Deanonymization

that X is simply a randomized identifier for *Alice* ($X = A_3$) and that she is nearby that particular relay's location. Later, at time t_i, *Eve* once again captures and relays a packet $\texttt{REQ}(X_i \rightarrow Y_i)$ to *Bob*. However, *Bob* does not respond with the hypothesized $\texttt{RSP}(B_i \rightarrow X_i)$. Therefore X_i != A_i and *Eve* determines *Alice* is not near that relay location.

The relay network allows *Eve* to maintain communication between devices at any distance, overcoming the limitations of M1. The static identity requirement is also lifted because an observation of successful communication patterns when relaying packets breaks the anonymity provided by identity randomization. The relay technique also supports packet replay if the communication protocol is vulnerable, allowing an attacker to replay a previously transmitted packet in various locations to observe the response and identify the victim without the need to capture and relay newer or synchronized packets. This occurs when the victim device does not check packet freshness using a sequence number or nonce.

3.4. Feasibility, Scope, and Impact

The IDBLEED hardware setup is relatively low-cost, making it feasible and practical even for a hobbyist and trivial for motivated nation-state actors. For passive attacks, an attacker simply sets up a wireless sniffer such as out-of-the-box BLE or Wi-Fi dongle, or a more advanced software defined radio setup. These can be used with Raspberry Pi's or smartphones with cellular data plans or nearby Wi-Fi, a common utility in public places such as coffee shops. An Internet connection allows transmitting data via an open-source publisher-subscriber message queue service and HTTP web application that an attacker can receive and analyze data. An attacker places these sniffing devices at any location their victim may be located. For active attacks, an attacker extends the passive setup to include a relay network of sniffing nodes at targeted locations that can also receive and re-transmit captured data packets. The effective range is limited by the wireless technology, however, the attacks remain practical given transmission rates and radio frequency waves' ability to travel through structures.

Wireless interfaces or dongles are available at technology retailers between \$10-\$50 USD, including those for BLE, Wi-Fi, and other IoT protocols such as Z-Wave, Zigbee, and LoRaWAN. Raspberry Pi's range between \$50-\$100 USD, with minimum RAM models being plenty sufficient. For a motivated nation-state actor, the setup cost becomes trivially low even when electing for advanced electronics, such as long-range antennas and signal amplifiers.

Example Attack Scenario & Workflow. The following is a generalized, example threat scenario to convey the real-world feasibility and practicality of the active IDBLEED attack. Consider *Alice*, a politician who *Eve* has targeted to track. *Eve* walks around *Alice*'s house when she is not home and sniffs packets from a previously paired smartspeaker broadcasting its presence via BLE beacons, since it is no longer within proximity and connected to *Alice*'s smartphone. Note, *Eve* does not need to observe the initial pairing between these exclusive-use devices. *Eve* then sets up and hides a smartphone relay node (R_1) outside of *Alice*'s home and within BLE range (approx. 30 feet) of the speaker. Additionally, *Eve* hosts a subscriber-publisher message queue web application that allows relay nodes to publish sniffed packets that are in turn multicasted to any other subscribing nodes. *Eve* deploys three additional smartphone relay nodes (R_2, R_3, R_4) at a coffee shop, the lobby at *Alice*'s work, and at her child's daycare. R_1 captures the smartspeaker broadcast packets and sends them to the message queue application, which R_2, R_3, and R_4 receive and in turn broadcast milliseconds later. *Alice*'s smartphone, on her person and currently at the coffee shop (R_2) near many other smartphones, replies to the broadcast, which R_2 captures and relays back to R_1. R_1 transmits the relayed smartphone response and determines the smartspeaker has a positive response. There are also numerous smartphones at $Alice's$ work and daycare, however, since none of these devices responded positively to the originally relayed packet from the smartspeaker, *Alice* can not be at these locations. Thus, *Eve* has determined *Alice*'s location to be the coffee shop. *This knowledge may then be used to launch more sophisticated cyber or other malicious attacks where knowing the victim's location and targeting the correct device is crucial.*

Attack Scope. To focus the scope of our IDBLEED attacks and research, we exclude tracking methods that directly obtain cryptographic keys through a pairing process explored by prior works BIAS [30], KNOB [31], and Downgrade [11]. Additionally, we exclude methods involving malware and instead focus purely on communication

Key	Length	Lifetime	Protection
Connection Signature Resolving Key (*CSRK*)	128 bit	∞	Verification
Long Term Key (*LTK*)	128 bit	Reset after pairing	Encryption
Session Key (*SK*)	128 bit	Every session	Encryption
Identity Resolving Key (*IRK*)	128 bit	∞	Authentication

TABLE 2. SUMMARY OF KEYS NEGOTIATED DURING PAIRING

protocol attacks. This helps underscore the flaws in the communication stages of the protocols themselves rather than rely on advanced malware capabilities that may track devices using GPS, logging, or other means.

Finally, we focus on scenarios where multiple devices are present at the location of the victim. This highlights the significance of exclusive-use communication patterns in the real world, since it would be significantly easier to identify a lone device at a specific location without using IDBLEED attacks. Further, the number of devices in an area does not impact the attack model or practicality since an attacker is observing the difference in communication patterns between trusted exclusive-use devices.

4. Concrete IDBLEED Attacks

4.1. IDBLEED in BLE Confidentiality

Our first analysis focuses on BLE encryption designed to provide data confidentiality. There are six types of keys exchanged during the BLE pairing process that create an association (Table 2). Four of these keys are used to encrypt data: Session Token Key (*STK*), Token Key (*TK*), Long Term Key (*LTK*), and Session Key (*SK*). However, previous works show *STK* and *TK* are insecure and vulnerable to key brute force attacks [32]. We therefore exclude those from our analysis and focus on *LTK* and *SK*, two keys used in *BLE Secure Connections* from SMP.

Protocol Workflow. *BLE Secure Connections* encrypts traffic bi-directionally using a derived *SK* that is unique to each connection. Two devices generate a valid *SK* by performing a process similar to Diffie-Helman, with real-world traces shown in Table 3, illustrated in Figure 4 , and further detailed in §C.

Attack Workflow. We accurately model Table 3 in Figure 5 to demonstrate IDBLEED during BLE encryption initiation for both passive and active methods.

- **Passive:** If *Eve* knows a peripheral (smartspeaker) is exclusive-use and observes it responding to a central (smartphone) with a `LL_START_ENC_RSP` message, rather than a `LL_REJECT_IND` message, they can deanonymize the central.
- **Active:** The active attack supports both relay and replay, modeled in Figure 5. At time t_1, *Eve* captures peripheral data packets, forwards them to other relay locations, and rebroadcasts near an anonymized smartphone using MAC address randomization. However, *Eve* deanonymizes the smartphone by observing the `LL_START_ENC_RSP` message. Replay is possible with *BLE Secure Connections*, as there is no sequence number and the first few packets used to initiate encryption are not encrypted. Note, to increase the covert power of the attack, *Eve* does not need to relay back the final `LL_START_ENC_RSP`, therefore reducing chances for *Alice* to be alarmed by a visual change in the user interface or connection notification.

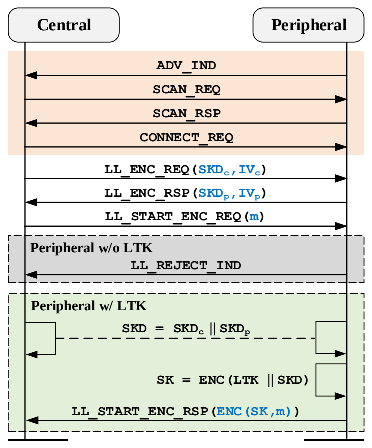

Figure 4. BLE Encryption Workflow

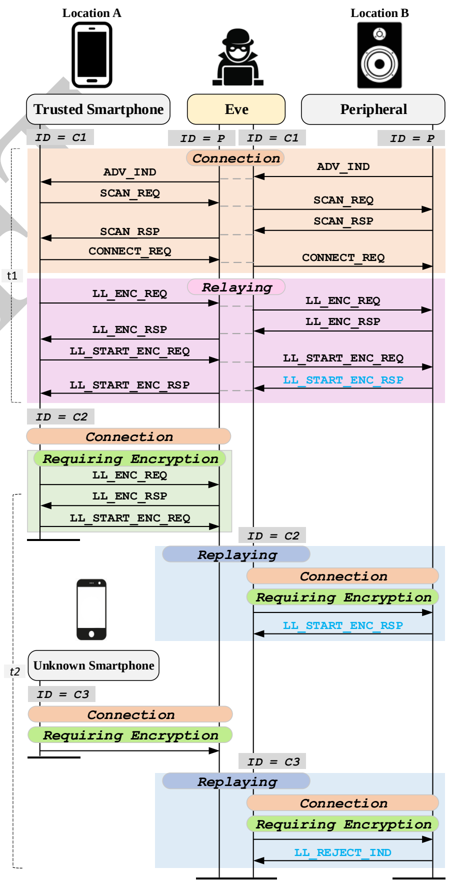

Figure 5. IDBLEED attacking BLE data encryption

Evaluation. This vulnerability exists due to the trusted association verification that uses an existing LTK and the communicating devices' MAC address. Therefore, all BLE devices that use LTK for security are vulnerable to this attack since it is a flaw in the BLE specification itself. We make two primary observations during our investigations: First, all of the evaluated peripheral devices use a static address—either a PA, which never changes, or an SRA, which only changes after a manual reset by the user. Second, we observe that different centrals may have different reactions when they receive a `LL_REJECT_IND` message from a peripheral—some devices terminate the connection after sending this message, while others simply continue waiting for further messages from the central.

Upon investigation we find both actions are acceptable based on the Bluetooth Core Specification 5.3, page 2844: "*The Link Layer of the Peripheral shall finalize the sending of the current Data Physical Channel PDU and may finalize the sending of additional Data Physical Channel PDUs queued in the Controller. After these Data Physical Channel PDUs are acknowledged, until this procedure is complete or specifies otherwise, the Link Layer of the Peripheral shall only send Empty PDUs,* `LL_TERMINATE_IND` *PDUs, and PDUs required by this procedure.*"

That is, the peripheral decides to continue the session by either sending empty PDUs to wait for responses from the central or terminate the session by sending a `LL_TERMINATE_IND` message. Finally, we observe that devices with functionality vital to other systems, such as keyboards, are more likely to terminate the connection.

No.	Time	Source ID	Destination ID	PDU Type
t0 = 0 min, C0 = `ad:d8:3e:a9:ba:52` (Passive attacker)				
1	00:00:36	58:d7:8e:c7:8e:31	Broadcast	ADV_IND
2	00:00:40	ad:d8:3e:a9:ba:52	58:d7:8e:c7:8e:31	SCAN_REQ
3	00:00:44	58:d7:8e:c7:8e:31	Broadcast	SCAN_RSP
4	00:00:48	ad:d8:3e:a9:ba:52	58:d7:8e:c7:8e:31	CONNECT_REQ
5	00:01:00	ad:d8:3e:a9:ba:52	58:d7:8e:c7:8e:31	LL_ENC_REQ
6	00:01:04	58:d7:8e:c7:8e:31	ad:d8:3e:a9:ba:52	LL_ENC_RSP
7	00:01:12	ad:d8:3e:a9:ba:52	58:d7:8e:c7:8e:31	LL_START_ENC_REQ
8	00:01:16	58:d7:8e:c7:8e:31	ad:d8:3e:a9:ba:52	LL_START_ENC_RSP
t1 = 15 min, C1= `be:a4:4e:dd:af:ee` (Active Attacker Using Relaying)				
101	00:15:36	58:d7:8e:c7:8e:31	Broadcast	ADV_IND
102	00:15:40	be:a4:4e:dd:af:ee	58:d7:8e:c7:8e:31	SCAN_REQ
103	00:15:44	58:d7:8e:c7:8e:31	Broadcast	SCAN_RSP
104	00:15:48	be:a4:4e:dd:af:ee	58:d7:8e:c7:8e:31	CONNECT_REQ
105	00:16:00	be:a4:4e:dd:af:ee	58:d7:8e:c7:8e:31	LL_ENC_REQ
106	00:16:04	58:d7:8e:c7:8e:31	be:a4:4e:dd:af:ee	LL_ENC_RSP
107	00:16:12	be:a4:4e:dd:af:ee	58:d7:8e:c7:8e:31	LL_START_ENC_REQ
108	00:16:16	58:d7:8e:c7:8e:31	be:a4:4e:dd:af:ee	LL_START_ENC_RSP
t1 = 30 min (Active Attacker Using Replaying)				
C2= `ae:f4:3f:d9:aa:12`				
201	00:30:36	58:d7:8e:c7:8e:31	Broadcast	ADV_IND
202	00:30:40	ae:f4:3f:d9:aa:12	58:d7:8e:c7:8e:31	SCAN_REQ
203	00:30:44	58:d7:8e:c7:8e:31	Broadcast	SCAN_RSP
204	00:30:48	ae:f4:3f:d9:aa:12	58:d7:8e:c7:8e:31	CONNECT_REQ
205	00:31:00	ae:f4:3f:d9:aa:12	58:d7:8e:c7:8e:31	LL_ENC_REQ
206	00:31:04	58:d7:8e:c7:8e:31	ae:f4:3f:d9:aa:12	LL_ENC_RSP
207	00:31:12	ae:f4:3f:d9:aa:12	58:d7:8e:c7:8e:31	LL_START_ENC_REQ
208	00:31:16	58:d7:8e:c7:8e:31	ae:f4:3f:d9:aa:12	LL_START_ENC_RSP
C3= `cf:ad:34:fe:ab:ee`				
211	00:30:36	58:d7:8e:c7:8e:31	Broadcast	ADV_IND
212	00:30:40	cf:ad:34:fe:ab:ee	58:d7:8e:c7:8e:31	SCAN_REQ
213	00:30:44	58:d7:8e:c7:8e:31	Broadcast	SCAN_RSP
214	00:30:48	cf:ad:34:fe:ab:ee	58:d7:8e:c7:8e:31	CONNECT_REQ
215	00:31:00	cf:ad:34:fe:ab:ee	58:d7:8e:c7:8e:31	LL_ENC_REQ
216	00:31:04	58:d7:8e:c7:8e:31	cf:ad:34:fe:ab:ee	LL_ENC_RSP
217	00:31:12	cf:ad:34:fe:ab:ee	58:d7:8e:c7:8e:31	LL_START_ENC_REQ
218	00:31:16	58:d7:8e:c7:8e:31	cf:ad:34:fe:ab:ee	LL_REJECT_IND

TABLE 3. AN EXCERPT OF REAL WORLD BLE PACKET TRACES. NOTE THAT `LL_START_ENC_RSP` IS ENCRYPTED.

4.2. IDBLEED via BLE Integrity

Although encryption offers stronger protection for data confidentiality, not all BLE devices support it. The BLE Connection Data Signing Procedure is an alternative to still provide both integrity and authenticity verification. A digital signature consisting of a counter and *MAC* is added to the end of the data payload. This allows one device to authenticate data sent between trusted devices signed with the unique *CSRK* originally shared during their pairing and association.

Protocol Workflow. The BLE Connection Data Signing Procedure includes a generation and verification stage, summarized as:

1) One of the two communicating devices, assumed here to be the Central denoted as c, generates a data signature for a given message (m) by concatenating it with a 32-bit self-increasing counter (*SignCounter*) to produce a new message (M). It then determines the length (L) of M.
2) The Central inputs the *CSRK* shared with its target Peripheral, the new message M, and length L into a message authentication code generation algorithm (defined in NIST Special Publication 800-3B [33]) to produce a 64-bit data signature (mac_c):

$$\begin{aligned} MAC_{64} &= CMAC(CSRK_{128} \mid\mid M \mid\mid L_{64}\ ) \\ &= CMAC(CSRK_{128} \mid\mid \\ &\qquad\qquad (\ m \mid\mid SignConter_{32}\ ) \mid\mid L_{64}\ ) \end{aligned}$$

3) The Peripheral receives m and mac_c and performs signature verification to determine if m is from a trusted Central by repeating steps 1-2 with its own locally stored *CSRK* to generate its own mac_p and compare with mac_c.

Attack Workflow. We present passive and active IDBLEED attacks on BLE data integrity using Figure 6, with details as follows:

Passive: *Eve* is able to observe the protocol sequence at any time and deanonymize the devices simply based on observing the responses due to the exclusive-use characteristic of the *CSRK* that is shared between and unique to the devices' pairing. For example, at time t_1, the smartphone uses a random BLE MAC address $C1$ (e.g., PRA or NPRA) to initiate a request $mac_c = CMAC(CSRK_{128} \mid\mid M \mid\mid L_{64}\ )$ to the smartspeaker, which it in turn responds with *ACK*. When the smartphone later changes its address to $C2$ and initiates another request to the smartspeaker, this communication is again observ-

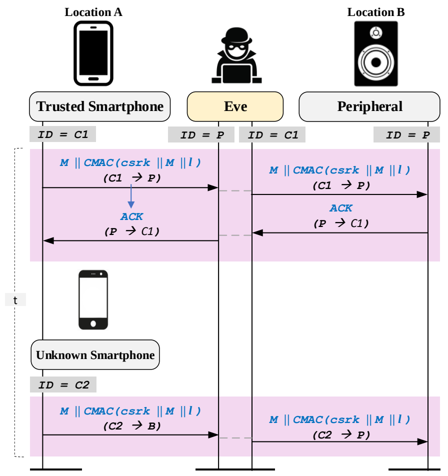

Figure 6. Side channel exploiting BLE data verification

able and *Eve* deanonymizes the smartphone $(C1 \mid\mid C2)$, regardless of the BLE address type being used.

Active: With a relay node near the smartspeaker, *Eve* is able to relay the mac_p. Devices near the target location nodes are unable to detect the relay and *Eve* is able to observe the presence of the ACK which serves as the single-bit information leak to deanonymize the smartphone. Note that due to the *SignConter* used in the message authentication code generation in steps 1-2, replay is not possible in this case study.

Evaluation. The nature of *CSRK*-based data integrity verification leaves all BLE peripherals, and by extension their paired centrals, that use it vulnerable to IDBLEED attacks. We validated this observation on various Android devices and found all are vulnerable, as highlighted in Table 4.

BLE peripherals are often firmware-defined, bare-metal IoT devices, such as an Apple AirTag. However, they can also be software-defined, which further widens the attack landscape and applicability. One implementation of software-defined peripherals is App-Defined Bluetooth Peripherals (AdBP) [34], which is provided by operating system APIs and used by mobile apps. A key feature of AdBP is the service protection provided by specifying permissions, of which we have identified two which use *CSRK* and signature verification that consequently leak the single-bit enabling IDBLEED: PERMISSION_ WRITE_SIGNED and PERMISSION_ WRITE_SIGNED_MITM. We note that iOS does not currently support using CSRK for their AdBPs which spares them from IDBLEED.

4.3. IDBLEED in Wi-Fi Authentication

Our next investigation looks closely at authentication in Wi-Fi Direct. Also known as Wi-Fi P2P, Wi-Fi Direct is a wireless networking protocol that allows devices to directly connect and communicate with each other without a separate AP. Devices discover each other, form a directly

Smartphone	Model	Chip	OS	BLE Version
Samsung	Galaxy S10	KM8D03042	Android 11.0	5.0
Google	Pixel 4	SM8150	Android 12.0	5.0
Google	Pixel 2	MSM8998	Android 9.0	5.0
Google	Pixel 4	SM8150	Android 10.0	5.0
HUAWEI	P10	BCM43455XKUBG	Android 9.0	4.2

TABLE 4. SUMMARY OF TESTED ANDROID DEVICES

linked network, and assign roles to orchestrate the temporary or persistent associations and communicate with each other using EAP for authentication.

Protocol Workflow. The Wi-Fi Direct connection and authentication protocol follows similarly to standard Wi-Fi with Probe and Association requests and responses, but adds an exchange of Invitation messages. One device initiates a connection and, once established, the devices form a group with one designated as the group owner (GO) that acts as an Soft-AP and the other as a client. The GO may specify a long-term configuration for the group, including a password for connections, while temporary groups are one-time use and do not have long-term configuration. Additional details provided in §D.

Attack Workflow. We provide Wi-Fi Direct packet traces with Table 5 and expand on IDBLEED attacks below.

- **Passive:** *Eve* can passively observe traffic at a location known to have a Wi-Fi Direct device owned by *Alice*, such as a home or office printer. The auto-connection may occur once a device comes in range, and if successful, *Eve* deanonymizes *Alice*.
- **Active:** The active IDBLEED attack involves capturing and relaying the probe and authentication packets to other locations. Upon observation of successful authentication, *Eve* deanonymizes *Alice*.

Evaluation. We observe that Wi-Fi Direct is vulnerable to both passive and active IDBLEED attacks due to the inherent pass or fail traffic patterns during authentication. In terms of practicality, Android devices have supported Wi-Fi Direct since version 4.0 and have provided various APIs for developers to customize Wi-Fi Direct groups. While the tested smartphones listed in Table 4 change their MAC address when joining a Wi-Fi Direct group, we find all of them are vulnerable to IDBLEED attacks. Note that as of iOS 7.0, iPhones do not use Wi-Fi Direct but rather their own direct link protocol known as *MultipeerConnectivity*, however similar connectivity resumption exists that reveal associations between devices [35].

4.4. IDBLEED in Wi-Fi Auto-Connection

Manually reconnecting to a Wi-Fi network can be a frustrating and time-consuming process. As a convenience feature, many devices send and receive probe automatically connect when they discover an in-range, known Wi-Fi or Wi-Fi Direct network. However, this feature can be exploited by attackers who create masquerade as previously associated

No.	Time	Source ID	Destination ID	Type
t0 = 0 min, C0 = `0e:8d:ae:c7:1e:50` (Passive Attacker)				
1	00:00:16	`0e:8d:ae:c7:1e:50`	`ff:ff:ff:ff`	`PROBE_REQ`
2	00:00:40	`12:df:a9:ef:fb:52`	`0e:8d:ae:c7:1e:50`	`PROBE_RSP`
3	00:00:44	`0e:8d:ae:c7:1e:50`	`12:df:a9:ef:fb:52`	`INVITATION_REQ`
4	00:00:48	`12:df:a9:ef:fb:52`	`0e:8d:ae:c7:1e:50`	`INVITATION_RSP`
5	00:00:54	`0e:8d:ae:c7:1e:50`	`12:df:a9:ef:fb:52`	`PROBE_REQ`
6	00:00:58	`12:df:a9:ef:fb:52`	`0e:8d:ae:c7:1e:50`	`PROBE_RSP`
7	00:01:00	`0e:8d:ae:c7:1e:50`	`12:df:a9:ef:fb:52`	`AUTH`
8	00:01:04	`12:df:a9:ef:fb:52`	`0e:8d:ae:c7:1e:50`	`AUTH`
9	00:01:12	`0e:8d:ae:c7:1e:50`	`12:df:a9:ef:fb:52`	`ASSOC_REQ`
10	00:01:16	`12:df:a9:ef:fb:52`	`0e:8d:ae:c7:1e:50`	`ASSOC_RSP`
t1 = 15 min, C1 = `0f:9e:fe:c2:2e:23` (Active Attacker Using Relaying)				
201	00:15:16	`0f:9e:fe:c2:2e:23`	`ff:ff:ff:ff`	`PROBE_REQ`
202	00:15:40	`12:df:a9:ef:fb:52`	`0f:9e:fe:c2:2e:23`	`PROBE_RSP`
203	00:15:44	`0f:9e:fe:c2:2e:23`	`12:df:a9:ef:fb:52`	`INVITATION_REQ`
204	00:15:48	`12:df:a9:ef:fb:52`	`0f:9e:fe:c2:2e:23`	`INVITATION_RSP`
205	00:15:54	`0f:9e:fe:c2:2e:23`	`12:df:a9:ef:fb:52`	`PROBE_REQ`
206	00:15:58	`12:df:a9:ef:fb:52`	`0f:9e:fe:c2:2e:23`	`PROBE_RSP`
207	00:16:00	`0f:9e:fe:c2:2e:23`	`12:df:a9:ef:fb:52`	`AUTH`
208	00:16:04	`12:df:a9:ef:fb:52`	`0f:9e:fe:c2:2e:23`	`AUTH`

TABLE 5. REAL-WORLD WI-FI PACKET TRACES. COLORS REPRESENT SCANNING, INVITATION, COMMUNICATION, AND RELAYABLE.

networks with the same SSID in order to trick devices into initiating a connection request. While this is well known as the EvilTwin attack [36], we provide a brief study and evaluation with a renewed perspective that focuses only on deanonymization without the goal for successful network connections that allow eavesdropping or data injection.

Protocol Workflow. The auto-connect workflow follows the standard Wi-Fi connection sequence as previously shown, but actively sends or receives probe beacons for nearby networks.

Attack Workflow. This attack is relatively simple to execute, as it does not require relaying packets between devices and is a hybrid of the passive and active IDBLEED attacks. *Eve* creates a Wi-Fi network using a name that is known to be in previously used by *Alice*, easily observed by capturing broadcast packets at a home or office. Upon observing a connection request, *Eve* has deanonymized *Alice*, or a member of a group she associates with.

Evaluation. The nuanced perspective we underscore is the ability to clearly observe a pass or fail condition in the authentication protocol. Since the device is set to auto-connect in many cases, this process occurs in the background without requiring *Alice* to accept and therefore is the essence of which the EvilTwin attack is based. While this attack can be easily carried out using a configured smartphone hotspot, *Eve* can also deploy nodes running custom software designed to stop the connection sequence once the single-bit leak is observed at the authentication stage.

5. Mitigation Framework

Having introduced the threats of our IDBLEED attacks, we now provide a generalized mitigation framework that serves as a blueprint for implementations with specific communication stacks. Specifically, we have developed and evaluated a simulation to demonstrate technical feasibility and benchmark performance impact of the framework's key features on an exemplar protocol baseline. To address core vulnerabilities in exclusive-use communication, we propose a new *Anonymization Layer* (AL) with the following goals.

G_1: **Removing Data Transmission Direction.** An anonymization layer must remove *Eve*'s ability to determine directionality of packets. To achieve this, even single-destination (unicast) packets must be sent over a communication protocol's broadcast channel or address while remaining addressable. Our solution generates, exchanges, and transmits various keys during the pairing and communication phases. These keys allow generating temporal transmission source identifiers that *Alice* appends to data destined for *Bob*. We propose and evaluate two solutions to achieve this goal that trade performance for frequency of anonymous identifier rotation: *Cache* or *Hash*.

- The *Cache* method has *Alice* and *Bob* use key information to separately calculate N source identities for each other. Upon receiving a packet, they linearly search pairing records to confirm their relationship. If a match is found, *Bob* knows the packet was bound for them specifically from *Alice* because the matching key was derived from their pre-shared secret keys unique to the pairing with *Alice* which acts as an *implicit destination address*, similar to the OKC used in 802.11 [37]. A temporal or counter interval parameter (I) is added to safeguard against replay attacks and long-term tracking techniques.
- The *Hash* method has *Alice* calculate a new resolvable source address for every packet to *Bob*, who then attempts to reproduce the same address using any of its paired record keys, similar to BLE's RRPA/IRK approach. [38]

In both cases, if a match is not found, the packet is simply discarded which prevents using unnecessary resources to process packets from unpaired devices. As an additional benefit, these methods add an ephemeral address randomization to otherwise static addresses.

G_2: **Removing Observable Packet Context & Entropy.** The next goal focuses on removing observable packet context or fields that indicate the packet type. We propose using counter-based encryption that uses rotating encryption keys using a pairing's pre-shared keys. These keys rotate with the determined interval to introduce additional entropy outside of that provided by the counter based encryption and avoids repeating patterns in common management or data packets, like methods used in [39]. Additionally, data packets are padded with a random number of bytes or to the full packet size to remove the ability to infer context based on size. [40]

G_3: **Adding Untrusted Responses.** Even with transmission direction and packet context anonymized, the single-bit side channel pattern still exists for protocols that exhibit no or a different response with a failure case. We mitigate this by randomly responding to untrusted entity requests. *Bob* randomly responds to untrusted requests using a random identifier and data bytes. The communication pattern between trusted and untrusted devices is now uncertain to *Eve*. Additionally, since the packet context is now unknown from G_2, *Eve* cannot be certain their replayed or relayed packets are aligned with the current protocol steps and state of the device.

G_4**: No Modifications (i.e., Transparent).** A generalized privacy preseving solution must not require modification to existing protocols, otherwise risk adoption challenges with compatibility. Our AL solution enhances communication protocol stacks by augmenting a new layer, encapsulating all layers above it to provide anonymity and privacy.

Formulas & Workflow. We now introduce the formulas, workflows, and process for AL. When *Alice* and *Bob* pair, they generate and exchange keys K_A and K_B of size s using a cryptographic random number generator (*CRNG*). They exchange keys and both create paired key PK_{AB} by XOR'ing K_A and K_B.

$$K_A \leftarrow CRNG(s), K_B \leftarrow CRNG(s) \tag{1}$$

$$PK_{AB} \leftarrow \text{XOR}(K_A,\ K_B) \tag{2}$$

Alice then uses an HMAC key derivation function (*HKDF*) to generate source key, SK_A, using PK_{AB}, and K_A, a static string, and output length L. *Bob* does the same, using K_B. Similarly, PK_{AB} and a different static string is used to create a shared paired encryption key PEK_{AB}.

$$SK_A \leftarrow HKDF(PK_{AB},\ K_A,\ \text{“}\texttt{ALsrc}\text{”},\ L) \tag{3}$$

$$PEK_{AB} \leftarrow HKDF(PK_{AB},\ \texttt{NULL},\ \text{“}\texttt{ALenc}\text{”},\ L) \tag{4}$$

The Cache and Hash methods now differentiate from this point. With the Cache method, *Alice* creates a set of N transmission keys TK_A ($\{TK_{A_0},\ TK_{A_1},\ ...,\ TK_{A_N}\}$) using SK_A and a time or counter interval I_{A_i} as parameters to a symmetric cryptographic function (*SENC*). The interval values for I can vary. For instance, an interval can be based on a time window if both devices have reasonably accurate system time. Otherwise, an initial seed value and counter mechanism may be used. Similarly, *Bob* creates transmission key set TK_B. Both *Alice* and *Bob* also repeat this for the other's set, each ending with TK_A and TK_B. The same is done to create a set of interval-based rotating encryption keys REK_{AB}:

$$TK_{A_i} \leftarrow SENC(SK_A,\ I_i) \tag{5}$$

$$REK_{AB_i} \leftarrow SENC(PEK_{AB},\ I_i) \tag{6}$$

Alice encrypts a message M destined for *Bob* by combining interval I_{A_i} and REK_{A_i} as input to a counter-based symmetric encryption algorithm (*CTR-SENC*):

$$M_E \leftarrow CTR\text{-}SENC\,(REK_{AB_i},\ I_i,\ M) \tag{7}$$

Alice now assembles an anonymized packet by concatenating TK_{A_i} with M_E and transmitting it over the transmission protocol's broadcast channel. *Bob* receives the packet and performs a lookup for TK_{A_i}. Upon successful match, *Bob* determines they're the packet's intended destination and it was generated by *Alice*, and retrieves the REK_{A_i} using the interval value to decrypt M_E.

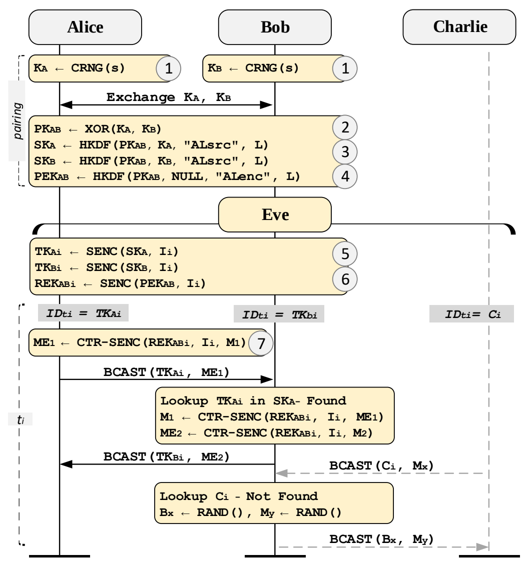

Figure 7. Passive Deanonymization AL Mitigation

Returning to the Hash method, *Alice* generates random bytes R_x of size l and combines them with SK_{AB} as arguments to an HMAC function to produce TKH_x. The R_x is concatenated with TKX_x to create the source transmission key TK_x which is appended to encrypted data like the Cache method. *Bob* receives the packet and takes R_x from TK_x and performs the same HMAC operation with each of its paired keys, SK_{AB}. With each attempt, *Bob* checks their created TKH_y against the remaining bytes of TK_x to potentially match TKH_x.

$$R_x \leftarrow CRNG(l) \tag{8}$$

$$TK_x \leftarrow R_x \mid\mid HMAC(R_x,\ SK_{AB}) \tag{9}$$

Security Analysis. With both methods, if there is no match, *Bob* discards the data and determines with probability P to reply with a random generated identifier and data. P can be tuned based on the broadcast channel's utilization, since there is no destination address, it is challenging to determine the intended recipient of anonymized packets. The mitigation provided by AL of passive IDBLEED attacks is illustrated in Figure 7 and follows the Cache workflow above.

The mitigation for active IDBLEED attacks is illustrated in Figure 8 and omits the workflow already reviewed for brevity. We examine a time t_i when *Eve* captures a packet at $LocationA$ from anonymous *Alice* with identifier X and relays it to *Bob*. When observing an encrypted response from *Bob*, *Eve* may assume they are paired and thus $X = A_i$. However, at the same time, *Eve* also captures a packet at $LocationC$ from anonymous $Charlie$ with identifier Y and relays it to *Bob*. Since *Bob* does not recognize identifier Y, they reply with random bytes as pseudo-communication. Given the exclusive-use characteristic, *Eve* must believe

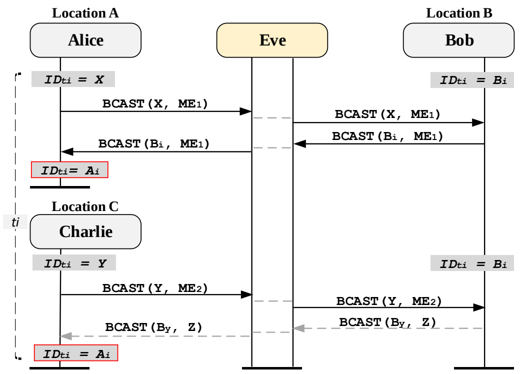

Figure 8. Active Deanonymization Mitigated by AL

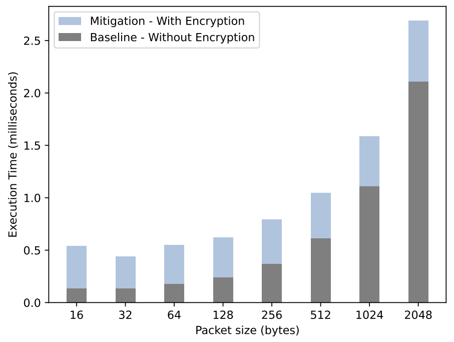

Figure 9. Encryption overhead on simulated protocol

Alice is at both locations, a logical contradiction, and therefore concludes the location of *Alice* can not be accurately or confidently determined.

Evaluation. We implemented both Cache and Hash AL methods and evaluated overhead impact on a simple exemplar protocol using Python 3.8. The simulation baseline is an application protocol which performs work to create bytes, incurs transmission or propagation delay, and processes the data upon receipt that varies in time with data size. We measured the overhead impact due to cryptographic and lookup operations relative to baseline during operation over varying sized packets and data sizes. We executed each experiment ten times, removing the lowest and highest values, and averaging the remaining eight data points gathered from execution on a bare metal Intel Xeon CPU at 2.20 GHz with 32 GB of RAM running Ubuntu 22.04.3.

Figure 9 shows average baseline and data packet encryption overhead for our simulated protocol sending and receiving 10,000 packets between two paired devices with data sizes incrementally doubling from 16 to 2048 bytes (B) to account for a wide variety of protocol maximum transmission sizes. The per packet execution time ranges are 1.976 milliseconds (ms) and 0.2765ms with means 0.610ms and 0.423ms for baseline and mitigation overhead, respectively. We observe the ratio of encryption and decryption overhead to our simulation baseline decreases with linearly increasing packet sizes. This indicates an initialization cost to encryption regardless of packet size and likely improved with

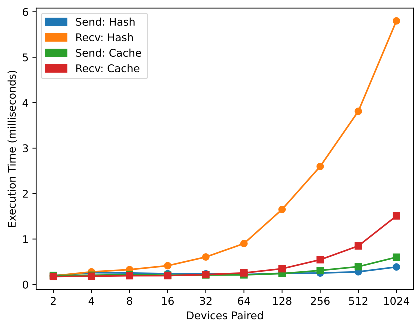

Figure 10. Pairing key lookup and resolution overhead

optimization in a real-world implementation. Typically IoT devices will transmit smaller packet sizes closer to the evaluated 32 and 64 B data size, which present a 0.321 mean ratio. Protocols with larger packet sizes between 1024 and 2048 B are more typical of enterprise networking environments see a decreased overhead per packet with 0.135 mean ratio.

Figure 10 shows the lookup and key resolution overhead for both the send and receive functions of the two Cache and Hash methods as paired devices doubles from from 2 to 1024. The execution time ranges are 0.192ms/5.611ms and 0.401ms/1.334ms with means 0.256ms/1.657ms and 0.279ms/0.447ms for sending/receiving Hash and Cache, respectively. We observe the overhead cost in a one-to-one exclusive scenario to be within 0.007ms and 0.014ms between the two methods for sending and receiving. However, the 0.034 range ratio for sending/receiving Hash transmissions indicates the lookup cost for this method increases significantly as paired device numbers increases.

Our evaluation shows AL as a viable mitigation framework to increase privacy and as an IDBLEED countermeasure. The destination anonymity (G_1) introduces less than 0.5ms mean overhead for one-to-one exclusive-use devices and up to 2.109ms or 6.185ms for Cache or Hash methods at 1,024 paired devices. The encryption step (G_2) introduces a relatively constant overhead at mean 0.423ms per packet.

6. Discussion

IDBLEED Practicality. As first mentioned in threat model introduction (§3.2), we assume an attacker can setup relays and the targeted devices are exclusive-use. With a known residence, and therefore location of their IoT peripheral devices, this becomes a practical exercise for a motivated attacker with standard sniffing equipment. Note, IDBLEED attacks do not identify arbitrary users and the active attack requires packets to be captured from an exclusive-use device already paired with the target device to deanonymize it. Fortunately, this also means the attacker does not need to be present for the initial pairing/bonding.

Alternatives to user tracking exist, but with additional technical challenges not present with IDBLEED. For instance, a camera can be used to record video in locations to determine *Alice's* presence. However, this approach requires high bandwidth and power to operate in addition to a either an advanced computer vision model or human in-the-loop to visually identify by inspecting the video feed. Further, the programmatic processing of packets and real-time notification offered by IDBLEED is lost. Additional challenges of optical-based approaches include relying on unobstructed field of vision, visual appearance discrepancies, and loss of quality at further distances which introduces unknown variability. Conversely, IDBLEED offers real-time, programmatic de-anonymizataion with approximately and minimally *1/n* confidence, where *n* is the number of paired devices in the exclusive-use relationship. We find exclusive-use is common for many IoT devices. Even if an IoT device is not designed to be used by a single user, it is likely to be used by a small group of associated people, such as a home appliance used by family members. In this scenario, IDBLEED could be used to track locations of a family member, although it would not be possible to differentiate between them. As such, IDBLEED can minimally be used to provide a data point in a multi-factor approach that combines more than one non-deterministic method, if not solely, to an overall confidence rating in user identification.

The IDBLEED active attack has the option of either relaying or replaying traffic, depending on the specifics of the wireless protocol. However, we believe that replay attacks are generally more practical than relay that requires both targeted devices to be able to process packets at the same time. This may not be possible if a device turns off its wireless interface in sleep mode to conserve power, as is the case with some wireless keyboards (or a small percentage of devices.) However, our research reveals that many stages of wireless protocols are still susceptible to replay attacks. For example, BLE devices that use an *LTK* to derive a *SK* for encrypting packets. While these encrypted packets include a Message Integrity Check (MIC) value to authenticate the sender and packet counters to prevent replay attacks, the initial packets that establish encryption (`LL_ENC_REQ` and `LL_START_ENC_REQ`) do not have these safeguards against replay attacks.

IDBLEED Impact & Consequences. IDBLEED offers a new option under the tracking attack class. As such, consequences are similar to other tracking attacks, but with the powerful discriminator that IDBLEED exploits an inherent flaw in the specifications of ubiquitous protocols that expose the single-bit side-channel in their communication patterns. Therefore, its exploitation window is significantly longer than other tracking attacks that exploit implementation flaws which often receive quick patch fixes. Tracking attacks such as IDBLEED pose a significant privacy and safety threat to victims. Tracking a person's location removes their autonomy to freely move about the world, particularly if their attacker is oppressive, or scenarios such as human trafficking, stalking, or pattern-of-life analysis of high-profile individuals. Knowing a person's location can be the initial link in a chain of more grave digital or physical attacks on the victim.

Anonymization Layer Practicality. Evaluation results of AL support its practicality for device-to-device communication anonymization. The relatively consistent cost for encryption signifies a startup cost that can be minimized via optimized, real-world implementations using C or assembly to run on low-cost microcontrollers that are common in IoT. The lookup computations can be optimized. For example, both Cache and Hash anonymity methods can store recently communicated paired records to decrease lookup times, similar to Most Recently Used (MRU) in memory management. Encryption performance can be increased through statically allocated memory buffers and algorithm optimization. Further, these optimizations are particularly realistic given our evaluation of minimal overhead with one-to-one or a handful of pairings that are exclusive-use devices. Additional research is necessary to prove out scalability for more dynamic and autonomous IoT systems that may intereact with thousands of new nodes per day, such as smart vehicles. For example, a public-key infrastructure paired with a real-time hierarchical spatial index to effectively act like an anonymized geo-fenced initial hash table lookup to decrease the number of keys to discover using Hash or Cache methods.

Anonymization Layer Security Analysis. The AL development goals (§5) were born from an adversarial mindset and its design ultimately aims to reduce the amount of meta and side-channel information available via sniffing. We accomplish this by removing transmission directionality to remove association, padding data to maximum MTU to remove packet size analysis, introducing entropy via encryption and counters to remove data pattern analysis, and replying to untrusted entities with pseudo-communication packets indistinguishable from those sent to trusted entities given the rotating source identifier and entropy in payload to remove the single-bit *exclusive-use* side-channel that we've introduced.

However, we recognize certain hardware may limit the feasibility of pseudo-communication and thus potentially be vulnerable to statistical analysis that considers responses amidst attacker-controlled packet flooding. To maintain primary functionality timing and quality of service expectations, device hardware may be overburdened to not be able to broadcast pseudo-responses. We first consider two scenarios that differ in number of nearby devices. The first, the attacker is near many devices that are also broadcasting with anonymized source address, thus making it difficult to determine if any target device's responses are with a legitimately trusted device or not. The second puts the attacker in a very controlled environment where they are absolutely sure there is only the target device present and therefore can statistically evaluate responses under massive flooding. However, this scenario is out-of-scope for anonymous communication solutions — if the attacker is sure the device is its target, there is no need to determine its location because that is inherently known. Between these two scenarios exists a spectrum of nearby device numbers, but then any results would require an advanced statistical treatment that results in an ambiguous confidence metric that must considered with many other factors. Effectively, AL discourages the attacker from this method and is left to pursue more reliable tracking methods.

Ethics and Responsible Disclosure. We considered ethics to the highest possible standard when performing our experiments. All experiments discussed in this paper were conducted in a controlled laboratory environment and no real-world devices outside of the lab were targeted. We disclosed the technical details of the BLE and Wi-Fi attacks, and potential countermeasures to the manufacturers of the tested devices, as well as the Bluetooth Special Interest Group (SIG). Several companies have acknowledged our findings, successfully replicated the attack, and plan to investigate this issue further.

7. Related Works

Privacy Attacks in IoTs. Previous efforts have discovered various ways by which IoT devices can be tracked through their wireless communication. These methods include the use of unique identifiers such as MAC addresses and IP addresses, as well as the analysis of side-channel information such as patterns in network traffic and power usage. For example, leaked information in encrypted traffic, such as the network connection frequency and the DNS server a device connects to, can be used to fingerprint IoT devices [12], or machine learning algorithms can be applied to network traces to identify IoT devices [13], [14], [15]. Huang et al. [16] demonstrated that attackers can infer user behaviors and lifestyles based on changes in the state of IoT devices and the associated network traffic.

Meanwhile, numerous Bluetooth device tracking attacks have been proposed, including those that collect advertising packets using sniffers [17], [18], [19], [6], [2], [20], [21], [3], [22], [5], [23], [11]. Previous approaches such as *BlueTrack* [3] and *BLEB* [22] track devices using public addresses, while Marco et al.[5] track classic Bluetooth devices that do not use address randomization by exploiting information leaked from frame encoding. Other attacks targeting specific implementations of Bluetooth devices include those for Apple devices [19], [6], [24], [25] and wearable fitness trackers [41]. Some attacks rely on static payloads to track devices, such as manufacture identifiers [2], [18], information elements [26], [27], and GATT attributes [42].

Tracking attacks have also been developed for Wi-Fi networks. Sapiezynski et al.[43] collected six months of human mobility data, including Wi-Fi and GPS traces recorded with high temporal resolution and found the time series of Wi-Fi scans contained a strong latent location signal that can be used for tracking. Scheuner et al.[28] developed a passive tracking system, *Probr*, that is manages various types of Wi-Fi capture devices and processes collected traces. In a separate study, Petre et al. [29] demonstrated the effectiveness of Wi-Fi tracking at large events exceeding 100,000 people over three days.

Our IDBLEED attack differs significantly from previous studies in that it exploits patterns observable from the trusted relationship between devices, rather than relying on static patterns in network traffic or power usage. While *BAT* attacks [7] were inspirational, our study differs significantly from theirs in the following ways.

Attack Vectors/Impact	IDBLEED	BAT
Generalized	✓	✗
Encryption	✓	✗
Authentication	✓	✗
Auto-Connection	✓	✗
Data-Verification	✓	✗
BLE	✓	✓
Wi-Fi	✓	✗
Replay	✓	✓
Relay	✓	✗
Hard to Patch	✓	✗
Highly Practical	✓	✗

TABLE 6. COMPARISON BEWEEN IDBLEED AND BAT ATTACKS.

Highlighted in Table 6, BAT authors only examined BLE allowlists and did not consider other side-channels such as authentication, data integrity, and encryption. Second, their work focuses solely on BLE, while we analyze attacks on BLE and Wi-Fi. Third, our single-bit side-channel abstraction is applicable far beyond any single protocol and extends to any communication protocol that has observable pattern differences between trusted and untrusted devices. While their mitigation focuses solely on a BLE vulnerability, our mitigation framework serves as a blue print for an entirely new anonymity layer. Finally, their study used replay attacks to track BLE devices, whereas we expand into relay attacks, an attack method that works even when the protocol is not vulnerable to replay attacks and is difficult to detect.

Anonymization Defense. Various solutions have been proposed for anonymous communication in IoT networks. Palmieri et al. [44] propose an anonymous routing framework between subnetworks which uses destination identifiers that an intermediary node uses to determine if the final hop has been reached for a device prior to broadcast. The Google/Apple Exposure Notification Framework [45] uses random rotating identifiers with payloads that are broadcasted and captured by nearby participating devices. Tor [46] uses onion routing for primarily TCP-based applications to provide an anonymous communication service consisting of nodes maintained by operators. However, these solutions require support from intermediate servers or gateway nodes to provide their privacy preserving functions. In contrast, our proposed AL framework exchanges keys and resolves communication directly on devices without the need for additional infrastructure or third-party configuration.

Zhang et al. propose *MASK* for mobile ad-hoc networks for anonymized single-hop communication [47] between grouped devices. It replaces traditional source and destination addresses at the MAC layer with ephemeral session and link keys, similar in concept to AL. However, their solution requires an explicit authentication and key exchange prior to a new session of data communication. Their approach is susceptible to IDBLEED, as this exclusive-use three-way handshake authentication mechanism is observable to either succeed or not given subsequent communication. Our solution with AL is transparent to other layers — there is no modification to existing layers and they retain established roles as they are today with authentication handled implicitly at AL by a successful lookup of a pairing key using the Hash or Cache method that is not observable to eavesdroppers.

8. Conclusion

We have shown that ubiquitous wireless communication protocols, such as BLE and Wi-Fi are vulnerable to deanonymization despite modern countermeasures such as identity randomization to prevent tracking attacks. The association between devices produces historically overlooked differences in observable traffic patterns that produce a single-bit side-channel leak, that we have termed *exclusive-use* devices, making them vulnerable to practical IDBLEED tracking attacks. We provide a generalized mitigation framework featuring our Anonymization Layer as a viable solution and quantify simulation results on performance overhead.

References

[1] S. Bluetooth, "Bluetooth core specification version 5.1," *Specification of the Bluetooth System*, 2019.

[2] J. K. Becker, D. Li, and D. Starobinski, "Tracking anonymized bluetooth devices," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 50–65, 2019.

[3] M. Haase, M. Handy *et al.*, "Bluetrack–imperceptible tracking of bluetooth devices," in *Ubicomp Poster Proceedings*, vol. 2, 2004.

[4] S. Bluetooth, "Bluetooth core specification version 4.2," *Specification of the Bluetooth System*, 2014.

[5] 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)*. IEEE, 2020, pp. 534–548.

[6] J. Martin, D. Alpuche, K. Bodeman, L. Brown, E. Fenske, L. Foppe, T. Mayberry, E. Rye, B. Sipes, and S. Teplov, "Handoff all your privacy–a review of apple's bluetooth low energy continuity protocol," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, pp. 34–53, 2019.

[7] Y. Zhang and Z. Lin, "When good becomes evil: Tracking bluetooth low energy devices via allowlist-based side channel and its countermeasure," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3181–3194.

[8] "Specifications | wi-fi alliance," https://www.wi-fi.org/discover-wi-fi/specifications, (Accessed on 12/07/2023).

[9] A. Development, "Implementing mac randomization," https://source.android.com/docs/core/connect/wifi-mac-randomization.

[10] iOS Development, "Wi-fi privacy," https://support.apple.com/guide/security/wi-fi-privacy-secb9cb3140c/web.

[11] Y. Zhang, J. Weng, R. Dey, Y. Jin, Z. Lin, and X. Fu, "Breaking secure pairing of bluetooth low energy using downgrade attacks," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 37–54.

[12] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," *arXiv preprint arXiv:1705.06805*, 2017.

[13] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "Profiliot: A machine learning approach for iot device identification based on network traffic analysis," in *Proceedings of the symposium on applied computing*, 2017, pp. 506–509.

[14] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis, "Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis," in *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2020, pp. 474–489.

[15] L. Yu, B. Luo, J. Ma, Z. Zhou, and Q. Liu, "You are what you broadcast: Identification of mobile and iot devices from (public) wifi," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 55–72.

[16] D. Y. Huang, N. Apthorpe, F. Li, G. Acar, and N. Feamster, "Iot inspector: Crowdsourcing labeled network traffic from smart home devices at scale," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 2, pp. 1–21, 2020.

[17] M. Jakobsson and S. Wetzel, "Security weaknesses in bluetooth," in *Cryptographers' Track at the RSA Conference*. Springer, 2001, pp. 176–191.

[18] K. Fawaz, K.-H. Kim, and K. G. Shin, "Protecting privacy of ble device users," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 1205–1221.

[19] G. Celosia and M. Cunche, "Discontinued privacy: Personal data leaks in apple bluetooth-low-energy continuity protocols," *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 1, pp. 26–46, 2020.

[20] ——, "Saving private addresses: an analysis of privacy issues in the bluetooth-low-energy advertising mechanism," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2019, pp. 444–453.

[21] A. Korolova and V. Sharma, "Cross-app tracking via nearby bluetooth low energy devices," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2018, pp. 43–52.

[22] T. Issoufaly and P. U. Tournoux, "Bleb: Bluetooth low energy botnet for large scale individual tracking," in *2017 1st International Conference on Next Generation Computing Applications (NextComp)*. IEEE, 2017, pp. 115–120.

[23] N. Ludant, T. D. Vo-Huu, S. Narain, and G. Noubir, "Linking bluetooth le & classic and implications for privacy-preserving bluetooth-based protocols," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021.

[24] M. Stute, A. Heinrich, J. Lorenz, and M. Hollick, "Disrupting continuity of apple's wireless ecosystem security: New tracking,{DoS}, and {MitM} attacks on {iOS} and {macOS} through bluetooth low energy,{AWDL}, and {Wi-Fi}," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3917–3934.

[25] 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, pp. 37–54.

[26] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms," in *Proceedings of the 11th ACM on Asia conference on computer and communications security*, 2016, pp. 413–424.

[27] H. Wen, Q. Zhao, Z. Lin, D. Xuan, and N. Shroff, "A study of the privacy of covid-19 contact tracing apps," in *International Conference on Security and Privacy in Communication Networks*, 2020.

[28] J. Scheuner, G. Mazlami, D. Schöni, S. Stephan, A. De Carli, T. Bocek, and B. Stiller, "Probr-a generic and passive wifi tracking system," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. IEEE, 2016, pp. 495–502.

[29] A.-C. Petre, C. Chilipirea, M. Baratchi, C. Dobre, and M. van Steen, “Wifi tracking of pedestrian behavior,” in *Smart Sensors Networks*. Elsevier, 2017, pp. 309–337.

[30] D. Antonioli, N. O. Tippenhauer, and K. Rasmussen, “Bias: Bluetooth impersonation attacks,” in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2020.

[31] D. Antonioli, N. O. Tippenhauer, and K. B. Rasmussen, “The {KNOB} is broken: Exploiting low entropy in the encryption key negotiation of bluetooth {BR/EDR},” in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1047–1061.

[32] M. Ryan, “Bluetooth: With low energy comes low security,” in *Proceedings of the 7th USENIX Conference on Offensive Technologies*, ser. WOOT’13. Berkeley, CA, USA: USENIX Association, 2013, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=2534748.2534754

[33] M. Dworkin, “Nist special publication 800-3b,” *NIST special publication*, vol. 800, no. 38B, p. 38B, 2005.

[34] Q. Zhao, C. Zuo, J. Blasco, and Z. Lin, “Periscope: Comprehensive vulnerability analysis of mobile app-defined bluetooth peripherals,” in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 521–533.

[35] “Multipeer connectivity | apple developer documentation,” https://developer.apple.com/documentation/multipeerconnectivity, (Accessed on 12/06/2023).

[36] D. A. Dai Zovi and S. A. Macaulay, “Attacking automatic wireless network selection,” in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*. IEEE, 2005, pp. 365–372.

[37] “Opportunistic key caching,” https://www.cisco.com/c/en/us/td/docs/wireless/controller/9800/17-2/config-guide/b_wl_17_2_cg/_opportunistic_key_caching.pdf.

[38] “Core specification | bluetooth technology website,” https://www.bluetooth.com/specifications/specs/core-specification-4-2/.

[39] A. Wang, C. Wang, X. Zheng, W. Tian, R. Xu, and G. Zhang, “Random key rotation: Side-channel countermeasure of ntru cryptosystem for resource-limited devices,” *Computers & Electrical Engineering*, vol. 63, pp. 220–231, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790617312740

[40] N. Doshi and D. Jinwala, “Constant ciphertext length in multi-authority ciphertext policy attribute based encryption,” in *2011 2nd International Conference on Computer and Communication Technology (ICCCT-2011)*, 2011, pp. 451–456.

[41] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, “Uncovering privacy leakage in ble network traffic of wearable fitness trackers,” in *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*. ACM, 2016, pp. 99–104.

[42] G. Celosia and M. Cunche, “Fingerprinting bluetooth-low-energy devices based on the generic attribute profile,” in *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things*, 2019, pp. 24–31.

[43] P. Sapiezynski, A. Stopczynski, R. Gatej, and S. Lehmann, “Tracking human mobility using wifi signals,” *PloS one*, vol. 10, no. 7, p. e0130824, 2015.

[44] “An anonymous inter-network routing protocol for the internet of things,” https://cora.ucc.ie/server/api/core/bitstreams/169fd0c8-2c3c-4446-9dc4-053b7e6c2d5f/content, (Accessed on 04/27/2024).

[45] “Exposure notification - cryptography specification.pages,” https://storage.googleapis.com/gweb-uniblog-publish-prod/documents/Exposure_Notification_-_Cryptography_Specification_v1.2.1.pdf, (Accessed on 04/27/2024).

[46] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” *Paul Syverson*, vol. 13, 06 2004.

[47] Y. Zhang, W. Liu, W. Lou, and Y. Fang, “Mask: anonymous on-demand routing in mobile ad hoc networks,” *IEEE Transactions on Wireless Communications*, vol. 5, no. 9, pp. 2376–2385, 2006.

Appendix

1. BLE Workflow

Expanded details from Figure 1.

1) A peripheral device, such as BLE-enabled audio earbuds, broadcasts data packets that include identifiable information such as its MAC address and Universally Unique identifier (UUID) to indicate its willingness to connect with another device.
2) A central device, such as a smartphone, discovers or recognizes a broadcasting peripheral device and begins to establish a connection.
3) The connection between devices is established.
4) The two devices optionally engage in pairing and bonding, which involves negotiating cryptographic keys such as a *Long Term Key* (*LTK*), *Connection Signature Resolving Key* (*CSRK*), or *Identity Resolving Key* (*IRK*), as presented in Table 2.
5) The central and peripheral devices communicate by reading or writing data to BLE attributes, key/value pairs that signify various functionalities.

2. Wi-Fi Workflow

Expanded details from Figure 1.

1) A Wi-Fi Access Point (AP), such as a router or a smartphone hotspot, broadcasts beacons containing its Service Set Identifier (SSID, i.e. network name) that are received by a client Wi-Fi device, such as a smartphone.
2) The user, or the device automatically, selects a network to connect to based on the SSID and any other desired criteria, such as the network type (e.g., private or public) or security settings.
3) The client and AP establish a connection through the exchange of association packets.
4) The client and AP authenticate each other through an authentication request and response process containing any necessary credentials.
5) With an established connection, the device and AP exchange data packets to complete the connection process and establish a reliable connection.

3. BLE Secure Connections Workflow

Expanded details from Figure 4:

1) The Central sends a Link Layer encryption request message (`LL_ENC_REQ`) containing a session key diversifier (SKD_c) and initialization vector (IV_c).
2) The Peripheral device receives the request, generates its own SKD_p and IV_p and includes them in an encryption response message (`LL_ENC_RSP`).
3) Both the central and peripheral combine parts of the shared SKD_c and SKD_p to create the SK. (The exchanged IV's are used by each recipient to initialize encryption.)

4) The central sends an unencrypted message (`LL_START_ENC _REQ`) and sets itself to receive encrypted data using the generated *SK*.
5) The peripheral responds with a message (`LL_START_ENC_ RSP`) encrypted using the *SK* and sets itself to receive encrypted data. If a valid SK cannot be generated due to missing a valid *LTK* shared during initial pairing, it will send a rejection message (`LL_REJECT_IND`) and may further terminate the the connection, depending on the implementation.
6) If the central receives the encrypted response message (`LL_ START_ENC_RSP`) from the peripheral, the two devices may begin transmitting and receiving encrypted data using the previously exchanged initialization vectors ($IV = IV_c \mid\mid IV_p$).

4. Wi-Fi Direct Workflow

1) A device broadcasts a probe request `PROBE_REQ` which is responded `PROBE_RSP` by another Wi-Fi Direct enabled device.
2) The initiating device sends a unicast invitation `INVITATION _REQ` to the responding device, which responds `INVITATION _RSP`.
3) The devices authenticate and associate with each other to form the group and assign roles.