Deanonymizing Device Identities via Side-channel Attacks in Exclusive-use IoTs & Mitigation

Christopher Ellis*, Yue Zhang†, Mohit Kumar Jangid*, Shixuan Zhao*, Zhiqiang Lin*
*The Ohio State University
†Drexel University

***Abstract*—Wireless technologies like Bluetooth Low Energy (BLE) and Wi-Fi are essential to the Internet of Things (IoT), facilitating seamless device communication without physical connections. However, this convenience comes at a cost—exposed data exchanges that are susceptible to observation by attackers, leading to serious security and privacy threats such as device tracking. Although protocol designers have traditionally relied on strategies like address and identity randomization as a countermeasure, our research reveals that these attacks remain a significant threat due to a historically overlooked, fundamental flaw in exclusive-use wireless communication. We define *exclusive-use* as a scenario where devices are designed to provide functionality solely to an associated or paired device. The unique communication patterns inherent in these relationships create an observable boolean side-channel that attackers can exploit to discover whether two devices "trust" each other. This information leak allows for the deanonymization of devices, enabling tracking even in the presence of modern countermeasures. We introduce our tracking attacks as IDBLEED and demonstrate that BLE and Wi-Fi protocols that support confidentiality, integrity, and authentication remain vulnerable to deanonymization due to this fundamental flaw in exclusive-use communication patterns. Finally, we propose and quantitatively evaluate a generalized, privacy-preserving mitigation we call ANONYMIZATION LAYER to find a negligible 2% approximate overhead in performance and power consumption on tested smartphones and PCs.**

I. INTRODUCTION

Wireless technologies like Bluetooth and Wi-Fi play a crucial role in the Internet of Things (IoT), enabling applications in smart homes, entertainment, healthcare, retail, and personal fitness. However, the wireless communication convenience also makes data exchanges vulnerable to sniffing attacks that compromise security and privacy, such as device location tracking. For example, smartphones using Bluetooth Low Energy (BLE) [1] to connect with accessories, such as earbuds, can have their packets observed by inexpensive sniffers. These packets contain a MAC address—*the smartphone's identifier*—which can be used to track the owner's location due to 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) recommends MAC address randomization, the periodic change of 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 leak static information used for tracking [5], [6], [2]. *BAT* attacks [7] also demonstrate BLE tracking 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 supporting BLE and Wi-Fi communication.

More specifically, we introduce the concept of *exclusive-use*, where a device exclusively provides functionality to a single or small group of trusted devices. This unique, trusted association allows for recognition and secure communication while preventing unauthorized access, even when devices change addresses. However, our research uncovers a *boolean side-channel leak* during the encryption, integrity verification, authentication, and auto-connection phases. By analyzing traffic patterns between exclusive-use devices—regardless of the protocol or randomization measures—attackers can deanonymize devices, enabling user tracking through the attack we introduce as IDBLEED.

Consider an attacker attempting to deanonymize Alice's smartphone, which uses address randomization. Alice leaves home and takes her smartphone but leaves behind her exclusive-use smartspeaker, which only accepts connection requests from her trusted device and rejects those from untrusted. The attacker deploys multiple relay nodes at different locations that forward packets between smartphones and the smartspeaker. If a distant smartphone communicates successfully with the smartspeaker through a relay, the attacker can infer that the smartphone belongs to Alice and identify its location, as communication would otherwise fail.

While IDBLEED attacks apply to any exclusive-use communication protocol, we evaluated two ubiquitous wireless technologies, BLE and Wi-Fi, for vulnerabilities related to privacy and tracking. For example, we found that BLE devices

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240703
www.ndss-symposium.org

using the *Connection Signature Resolving Key* (*CSRK*) for data integrity verification are vulnerable. Although the *Long Term Key* (*LTK*) ensures data confidentiality, it also exposes a boolean side-channel due to different communication patterns between trusted and untrusted devices.

Devices rely on trust mechanisms to provide functionality, which inherently creates exclusive-use patterns and allows attackers to break device anonymity. A sufficient countermeasure requires protocol designers to yet again consider the classical dilemma of balancing security and privacy with performance and functionality. We propose our ANONYMIZATION LAYER (AL), a generalized mitigation that allows devices to communicate with trusted peers while obscuring exclusive-use characteristics. This layer eliminates sniffable source-destination information through implicit addressing and encrypts packets to prevent side-channels throughout the communication phases after they have formed their trusted relationship. We evaluate the performance and power consumption impact against a baseline on a multi-core PC laptop and Google Pixel 7, finding negligible overhead.

Contributions. Our contributions in this paper are threefold:

1) **Novel Vulnerability (§III).** We are the first to demonstrate the vulnerability in a ubiquitous wireless communication scenario we call *exclusive-use*, where distinct traffic patterns at specific stages reveal trusted relationships. We focus on IoT devices and show that this fundamental and overlooked flaw can be exploited by attackers through passive observation of wireless traffic or by actively relaying or replaying packets.
2) **Concrete Attacks (§IV).** We confirm through protocol and real-world packet analysis that widely used wireless technologies, including BLE and Wi-Fi, are vulnerable to tracking attacks that exploit exclusive-use characteristics to deanonymize devices—an attack we introduce as IDBLEED. Further, these attacks are feasible at low-cost, leveraging protocol traffic pattern vulnerabilities without requiring sophisticated device compromise or malware.
3) **Mitigation Solution (§V).** We propose a novel generalized mitigation that introduces ANONYMIZATION LAYER (AL) which supports anonymous communication between devices over broadcast channels using ephemeral identifiers, removes the need for destination addresses, and addresses the boolean side-channel leak through pseudo-communication with untrusted devices. We implement AL in C and evaluate its performance overhead on-device and observe a negligible 1.808 or 2.038% mean overhead, for PC or smartphone respectively, measured from 1.4 million packets varying in data size between 16 and 2048 bytes. Additionally, our key resolution method (*Cache*) outperforms existing methods by 2.5x-40.4x (PC) and 1.5x-30.8x (smartphone) performance increase, measured over 1 to 512 pairs.

II. BACKGROUND

Wireless technologies are essential to IoT, freeing devices from the constraints of physical wiring by enabling communication and exchange data over the air. Although various wireless protocols exist, this paper focuses on BLE [1] and Wi-Fi [8] due to their ubiquity and widespread adoption, even beyond IoT. We present background information below.

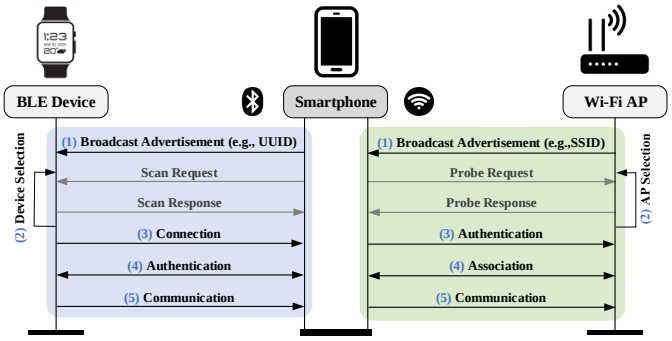

Fig. 1: Typical BLE and Wi-Fi workflows.

A. Wireless Protocols

Bluetooth Low Energy. BLE enables devices to establish bi-directional wireless links, commonly found in wireless speakers, car infotainment systems, smart home devices, and smartwatches. With a similar transmission range and lower power requirements than classic Bluetooth, BLE is very common in IoT devices [9], [10]. With each connection, BLE devices assume either a central or peripheral role, defining their protocol responsibilities during communication. The general workflow of BLE communication, illustrated in Figure 1, has up to five stages:

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 II.
5) The central and peripheral devices communicate by reading or writing data to BLE attributes, key/value pairs that signify various data and functionalities.

Wi-Fi. Wireless Fidelity (Wi-Fi) is a family of protocols based on IEEE 802.11 standards that enable devices to connect to networks wirelessly. Wi-Fi Access Points (AP) act as servers, allowing client devices—such as computers, smartphones, and smart devices—to connect to the network using various authentication mechanisms, as shown in Figure 1. Other Wi-Fi protocols, such as Wi-Fi Direct, operate similarly to classic Wi-Fi but enable traditionally client devices to act

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 I: Summary of four types of BLE addresses

as APs. A common use case is smartphones creating ad-hoc networks to exchange data without using a cellular service. The general workflow of Wi-Fi is as follows:

1) A Wi-Fi 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.

B. 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, included in a transmitted packet to act like a recipient's name on a mail envelope.

Unfortunately, MAC addresses are inherently vulnerable to sniffing when packets are sent over the air using a wireless technology. This allows attackers to identify and track devices, raising user privacy concerns. For example, an attacker can sniff packets exchanged between a paired smartwatch and smartphone that may use unique MAC addresses and determine when a victim arrives or departs a particular location.

As a countermeasure, some devices now use address randomization to enhance privacy and defend against tracking while still allowing addressable devices. Typically this involves a protocol that periodically generates new, random identifiers, while still allowing communication with trusted devices. The randomization technique used by a real-world device is determined primarily by user settings, developers providing backwards compatibility, or technology limitations.

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 I.

- *Public Address* (PA): globally static, manufacturer assigned to serve as a unique device identity. PA is vulnerable to MAC address tracking attacks, as it never changes.
- *Static Random Address* (SRA): randomly 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): randomly generated, periodically device-created, and resolvable by paired devices with a shared *Identity Resolving Key* (IRK).
- *Non-Resolvable Private Address* (NRPA): randomly and periodically generated, but is never intended to be resolvable depending on the implementation.

Wi-Fi Address Randomization. Wi-Fi MAC address randomization is widely used in recent Android [11] and iOS [12] versions. Android uses two methods: the first generates a fixed random MAC address per network based on its AP beacon attributes, preventing tracking across networks. The second generates a new MAC address every 24 hours if the device disconnects, defending against traffic sniffing. iOS 14+ similarly assigns random MAC addresses per network. Additionally, Wi-Fi probe requests use random addresses to prevent tracking during AP discovery. To ensure AP compatibility, both Android and iOS may use Wi-Fi authentication protocols like Extensible Authentication Protocol (EAP), where the device proves its identity through challenge-response exchanges.

C. Protocol Features

Confidentiality. Encryption ensures data confidentiality for network protocols. During communication, devices negotiate secret keys for verification or encryption. For example, BLE devices may exchange *LTK* during pairing, later deriving encryption session keys. The same *LTK* is used to generate identical session keys, securing the trusted relationship.

Integrity. Data signatures or message authentication codes (MAC) using shared cryptographic keys ensure data integrity—confirming the data matches the originator's intent. For example, BLE verifies unencrypted data with a *CSRK*, exchanged during pairing. The key generates a MAC appended to the message as packet payload. Each BLE device holds a unique *CSRK* with others, ensuring exclusive-use and preventing unauthorized alterations.

Authenticity. Authentication verifies device identity for network or resource access, often using a challenge-response protocol with a shared cryptographic key. One device sends a challenge with a MAC, and the other responds by solving it. If valid, the first device authenticates the responder. While often paired with encryption, authentication serves a distinct role.

Auto-Connection. Auto-connecting to trusted networks or devices improves user experience. Smartphones automatically join familiar Wi-Fi networks, like home routers, when detected via probes. Similarly, IoT apps connect with paired BLE devices when nearby. This seamless background connectivity spares users from manual selection and delays.

III. IDBLEED TRACKING ATTACKS

A. Deanonymization via Exclusive-use

We define *exclusive-use* as the observable association between two or more trusted devices based on their communication patterns. An exclusive-use device interacts distinctively with a trusted peer compared to other devices. This common protocol design mitigates security threats by restricting access to specific resources and services to devices with an established trust relationship, typically owned by the same user or associated group.

However, we observe that the communication patterns of this design also inherently deanonymize devices, enabling attackers to track victims. While numerous studies have investigated tracking attacks [2], [3], [5], [6], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], we are confident that we are the first to explore and raise awareness of this historically overlooked and nuanced characteristic fundamental to exclusive-use devices. Key distinctions from related work are discussed in §VIII. *We further recognize that numerous protocols exhibit exclusive-use characteristics, making our concept not only applicable to IoT but also ubiquitous across communication technologies.*

Effectively, a device's exclusive-use characteristics are observed via a boolean traffic pattern 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. Note, we use "boolean" to describe a binary, success/failure, yes/no, if/else, etc., traffic pattern, and not an actual data bit in the packet or a boolean datatype.

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.

B. Scope

Attack Scope. We focus the scope of our IDBLEED attacks and research by excluding tracking methods that directly obtain cryptographic keys through a pairing process, explored by prior works *BIAS* [32], *Ghost Attacks* [33], *Downgrade* [13], and *KNOB* [34]. Further, we do not investigate the initial pairing process, which has a low or one-time occurrence rate and is difficult to time the capture. Additionally, we exclude methods involving malware and instead focus purely on communication protocol attacks. This underscores the flaws in the protocol communication stages themselves rather than rely on advanced malware capabilities that may track devices using GPS, logging, or other means. We primarily focus on ubiquitous wireless communications technologies to include BLE and WiFi that demonstrate the vulnerability, without augmentation with other techniques that require fields of view, sound localization techniques, etc. Packet collection via sniffing is constrained by the target device's communication protocol and transmission power, however, does not limit the total distance a packet may ultimately travel between devices via other technologies and relay methods.

Victim Scope. Our research focuses on scenarios where multiple devices are present at the potential 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. Our research investigates exclusive-use patterns in known, trusted relationships and does not target arbitrary victims, therefore known potential device location of (residence, office, etc.) is required.

C. Threat Model

Attacker Goals & Motivations. An attacker is a person or group motivated to track a specific, targeted person's location. Some examples include family members or friends by personal investigators; political figures, government employees, researchers with sensitive information, or people in power by nation-states or radical extremists; high-net worth individuals or celebrities by paparazzi, stalkers, or extortionists; people involved in legal cases such as defendants, victims, informants, or lawyers by aggressors with opposing views. Example locations include a home, office, daycare, coffee shop, airport, state, country, facility, etc. Motivations include tracking locations or pattern-of-life analysis as a step in a larger goal to potentially inflict physical harm, harass, extort, gather intelligence, influence, or exploit in ways outside of the digital world but first requires knowledge of a victim's physical location. Note, there is no age discrimination of the victims for tracking attacks. The potential attackers and their motivations for tracking victims is nearly endless and alarming, underscoring the need for a sufficient defense that we provide in §V.

Real-world Attacker Feasibility. 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 web application that an attacker can remotely receive and analyze data. An attacker places these sniffing nodes at any location their victim may or soon 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

between victim devices and nodes, however, the attacks remain practical given radio frequency waves' ability to travel through structures and transmission speeds in an Internet-connected relay network.

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 cost becomes trivially low even when electing for advanced electronics, such as long-range antennas and signal amplifiers.

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

1) The victim's devices are exclusive-use and previously formed a trusted relationship. This implies a unique association between the user and the devices, which exhibit distinct success and failure communication patterns.
2) The attacker can sniff and potentially relay and retransmit communication traffic between devices. Note, sniffing an initial "bonding" process is not assumed or required.

D. Attack Method Workflows

We now present the generalized Passive (M1) and Active (M2) 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, relays, and re-transmits their communication traffic. These workflows provide a basis for our real-world analysis in §IV.

(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 boolean 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 becomes unsuitable if they move beyond their communication protocol's range—such as when a smartphone is taken to a coffee shop while a BLE-paired smart speaker remains at home. Further, the passive attack relies on one device maintaining a static identity, unless communication is observed during asynchronous identity randomization between

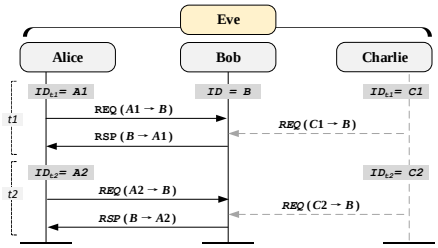

Fig. 2: Passive Deanonymization

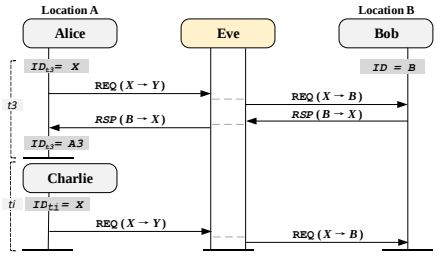

Fig. 3: Active Deanonymization

the two devices, as explored in [7]. These limitations are addressed by 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 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 node location.

The relay network enables *Eve* to facilitate communication between devices at any distance, bypassing the proximity limitations of M1 and discussed in attack scope. The two devices remain unaware that packets are being relayed, given absence of a distance or time-bounding protocol. The active attack eliminates the static identity requirement of M1 since observing successful communication patterns while relaying packets breaks the anonymity from identity randomization. Additionally, the relay technique allows for packet replay if the communication protocol is vulnerable (e.g., not verifying packet freshness using a sequence number, nonce, TTL, etc.), enabling an attacker to replay previously transmitted packets at different locations to identify the victim.

E. Real-world Attack Scenario & Workflow Example

The following threat scenario illustrates the feasibility of the active IDBLEED attack. A nation-state actor, *Eve*, targets politician *Alice* to track her movements. *Eve* learns

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 II: Summary of keys negotiated during pairing

Alice's home address and sniffs BLE packets from her paired smartspeaker while nearby, which is not currently connected to *Alice's* smartphone and without needing to observe the initial bonding. *Eve* sets up a smartphone relay node (R_1) outside *Alice's* home, within BLE range (about 30 feet). She also hosts a web application that allows relay nodes to publish sniffed packets to other subscribers. Next, *Eve* deploys three additional relay nodes (R_2, R_3, R_4) at a coffee shop, *Alice's* workplace, and her child's daycare. R_1 captures the smartspeaker's broadcast packets, relaying them to R_2, R_3, and R_4, which they immediately broadcast over BLE. While at the coffee shop (R_2), *Alice's* smartphone replies to the broadcast, which R_2 captures and relays back to R_1. R_1 then confirms the smartspeaker received a positive response. Since none of the smartphones at *Alice's* work or daycare (R_3, R_4) responded, *Eve* concludes that *Alice* is at the coffee shop (R_2). This confirmation of location can now be leveraged for more sophisticated cyber or malicious attacks.

IV. Concrete IDBleed Attacks

A. IDBleed *in BLE Confidentiality*

Our first analysis focuses on BLE encryption that provides data confidentiality. There are six types of keys exchanged during the BLE pairing process that create an association (Table II). 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 [35]. We 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 III, illustrated in Figure 4, and further detailed as follows:.

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*.

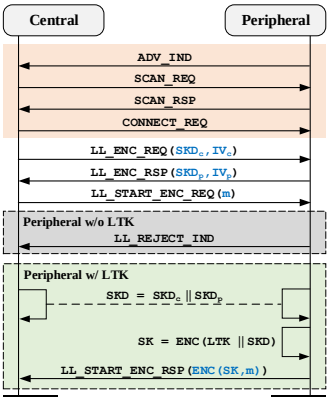

Fig. 4: BLE Encryption Workflow

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 ($IV = IV_c \mid\mid IV_p$).

Attack Workflow. We accurately model Table III in Figure 5 to demonstrate IDBleed during BLE encryption initiation.

- **Passive:** *Eve* knows a peripheral (smartspeaker) is exclusive-use and observes it responding to a central (smartphone) with a `LL_START_ENC_RSP` message, versus a `LL_REJECT_IND` message, they deanonymize the central.
- **Active:** Supports relay and replay, modeled in Figure 5. At time t_1, *Eve* captures peripheral data packets, forwards them to relay locations to rebroadcast near an anonymized smartphone using MAC address randomization. *Eve* observes the `LL_START_ENC_RSP` message and deanonymizes the smartphone. Replay is possible with *BLE Secure Connections*, as there is no sequence number and the first few packets to initiate encryption are not encrypted. To increase the attack's covertness, *Eve* doesn't need to relay back the final `LL_START_ENC_RSP` thus reducing a chance *Alice* is alarmed by visual user interface notifications.

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—it is a flaw in the BLE specification itself. We make two primary observations: First, all of the evaluated peripheral devices use a static address—either a PA, which never changes, or an SRA, which

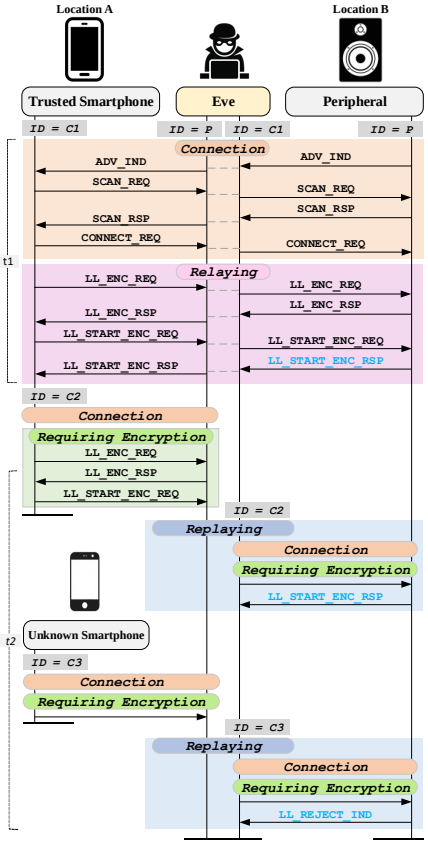

Fig. 5: IDBLEED attacking BLE data encryption

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 III: Real-world BLE traces w/ encrypted LL_START_ENC_RSP

only changes after a manual reset by the user. Second, we observe different centrals may have different reactions when they receive a peripheral's LL_REJECT_IND message—some peripherals terminate the connection after sending this message, while others wait for new 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 devices vital to other systems, such as keyboards, are more likely to terminate the connection.

B. 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 Message Authentication Code (*MAC*) is appended to 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:

1) One of two communicating devices, assumed here to be Central c, generates a data signature for a message (m) by concatenating a 32-bit self-increasing counter (*SC*) to produce a new message (M) and determines its length L.
2) The Central inputs the *CSRK* shared with its target Peripheral, M, and L into a message authentication code generation algorithm (defined in NIST Special Publication 800-3B [36]) to produce a 64-bit data signature mac_c:

$$\begin{aligned} MAC_{64} &= CMAC(CSRK_{128} \,||\, M \,||\, L_{64}) \\ &= CMAC(CSRK_{128} \,||\, (\, m \,||\, SC_{32}\,) \,||\, L_{64}) \end{aligned}$$

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 detail passive and active IDBLEED attacks on BLE data integrity using Figure 6 and the following:

- **Passive:** *Eve* observes the protocol sequence at any time and deanonymizes the devices by observing the commu-

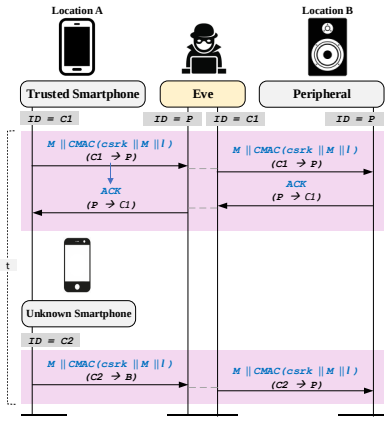

Fig. 6: Side channel exploiting BLE data verification

nication pattern due to the devices' shared and unique *CSRK*. 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) to the smartspeaker, which it responds with *ACK*. Later, the smartphone changes its address to $C2$ and initiates a request to the smartspeaker—this traffic is again observable and *Eve* deanonymizes the smartphone, regardless of the BLE address type.

- **Active:** With a relay node near the smartspeaker, *Eve* is able to relay the mac_p. Devices near the target location nodes are unaware of the relay and *Eve* observes the *ACK* that deanonymizes the smartphone. Note, due to the *SC* used in the *mac* generation in steps 1-2, replay is not possible.

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

BLE peripherals are often firmware-defined, bare-metal IoT devices, such as Apple AirTags. However, they can also be software-defined, widening IDBLEED's attack landscape and applicability. One example is App-Defined Bluetooth Peripherals (AdBP) [37], provided by operating system APIs and used by mobile apps. A key feature of AdBP is service protection via permissions—we have identified two that use *CSRK* and signature verification, consequently leaking the boolean side-channel: `PERMISSION_ WRITE_SIGNED` and `PERMISSION_ WRITE_SIGNED_MITM`. Note, iOS is spared as it doesn't currently support *CSRK* in their AdBPs.

C. IDBLEED *in Wi-Fi Authentication*

We now look closely at authentication in Wi-Fi Direct (also known as Wi-Fi P2P), a wireless networking protocol that allows devices to directly connect and communicate with each other without a separate, dedicated AP. Devices discover each other, form a directly linked network, and assign roles to orchestrate the temporary or persistent associations and communicate with each other using EAP for authentication.

Smartphone	Model	Chip	OS	BLE Ver.
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 IV: Summary of tested Android devices

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 a 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. The workflow is as follows:

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.
4) The devices can now communicate with each other.

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

- **Passive:** *Eve* observes traffic at a location known to have a Wi-Fi Direct device owned by *Alice*, such as a home or office printer. Auto-connection may occur once a device comes in range and deanonymizes *Alice*, if successful.
- **Active:** *Eve* captures and relays probe and authentication packets to other locations. Upon observation of successful authentication, *Alice* is deanonymized.

Evaluation. We observe that Wi-Fi Direct is vulnerable to both passive and active IDBLEED attacks due to the inherent pass/fail traffic patterns during authentication. While the tested smartphones listed in Table IV change their MAC addresses upon joining a Wi-Fi Direct group, we find that all of them remain vulnerable to IDBLEED attacks. Note that as of iOS 7.0, iPhones do not use Wi-Fi Direct but instead their own direct link protocol known as *MultipeerConnectivity*. However, similar connectivity resumption mechanisms exist that reveal associations between devices [38].

D. IDBLEED *in Wi-Fi Auto-Connection*

Many devices send and receive probes to auto-connect upon discovering an in-range, known Wi-Fi or Wi-Fi Direct network. This feature can be exploited by attackers who masquerade as previously associated networks with the same SSID in order to trick devices into initiating a connection request. While this is well known as the *EvilTwin* attack [39], we provide a brief study and evaluation with a renewed

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 V: Real-world Wi-Fi packet traces. Colors represent scanning, invitation, communication, and relayable

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 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 an SSID known to be previously used by *Alice*, easily observed by capturing broadcast packets at a home or office. *Eve* deanonymizes *Alice* upon observing a connection request.

Evaluation. The nuanced perspective we underscore is the ability to clearly observe a boolean condition in the authentication protocol. Since devices are set to auto-connect in many default cases, this process occurs in the background without confirmation from *Alice*—the essence that *EvilTwin* is based. While this attack can be easily executed using a configured smartphone hotspot, *Eve* can also deploy nodes running custom software designed to stop the connection sequence once the boolean leak is observed at the authentication stage for more covert tracking.

V. MITIGATION DESIGN

Having introduced the threats of our IDBLEED attacks, we now provide a generalized mitigation solution for implementation and integration with specific communication stacks. More specifically, we have developed an anonymized protocol to demonstrate technical feasibility and evaluate performance impact of our solution's key features as a privacy-enhancing technology that addresses the core vulnerabilities in exclusive-use communication. We introduce and propose ANONYMIZATION LAYER (AL) with the following goals (G_1 - G_4).

G_1: Remove Data Transmission Direction. A valid mitigation must prevent *Eve* from determining packet direction through flow analysis. This requires sending originally unicast packets over a protocol's broadcast channel while keeping them addressable. Our solution generates, exchanges, and transmits keys during pairing and communication, enabling temporal source identifiers that *Alice* appends to data destined for *Bob*. We propose two solutions for key resolution: *Cache* and *Hash*—balancing performance with anonymous identifier rotation frequency.

The *Cache* method allows *Alice* and *Bob* to use keys to calculate N source identities for each other. Upon receiving a packet, they linearly search pair records. If a match is found, *Bob* knows the packet was intended for them, and from *Alice*, as the key derives from their unique pre-shared secret, acting as an *implicit destination address*, similar to the OKC in 802.11 [40]. A temporal or counter parameter (I) protects against replay attacks and long-term tracking.

The *Hash* method requires *Alice* to calculate a new resolvable source address for each packet to *Bob*, who reproduces the same address using any of its paired record keys, similar to BLE's RRPA/IRK approach [41].

For both, if no match is found, the packet is discarded to conserve resources. These methods also provide ephemeral address randomization for otherwise static addresses.

G_2: Remove Observable Packet Context & Entropy. The second goal removes observable packet context or fields that indicate packet type. We propose counter-based encryption with rotating keys, which change at set intervals to introduce extra entropy and avoid repeating patterns in management or data packets, as in [42]. Packets are padded with random bytes or to full size, preventing size-based context inference [43].

G_3: Add Pseudo-responses. Even with anonymized transmission direction and packet context, boolean side-channel patterns still exit in protocols with differing success/failure responses. Our mitigation has *Bob* randomly send pseudo-responses to untrusted packets using random identifiers and data, creating uncertainty for *Eve* regarding communication patterns. Additionally, since packet context is unknown due to G_2, *Eve* cannot determine with certainty if their replayed or relayed packets align with the current protocol steps and device state.

G_4: No Modifications (i.e., Transparent). A generalized privacy-enhancing solution should not require modifications to existing protocols to avoid adoption challenges. Our AL solution enhances communication protocol stacks with a new layer, encapsulating all layers above it to provide anonymity. This enables existing layers to operate without knowledge of AL while preserving their functionality.

Defense Scope. AL aims to sufficiently achieve the above design goals, but the scope does not include replacing entire layer functionality—it augments existing communication stacks agnostically. It also does not shape traffic timing as the throughput impact is specific to application timing requirements and hardware capability. Further, since it is transparent to the other layers, it does not aim to provide replay or relay defense.

Formulas & Workflow. *Alice* and *Bob* begin pairing by

generating and exchanging keys K_A and K_B of size s using a cryptographic random number generator (*CRNG*). They then 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}. Note, this process is one-time and secure using established encrypted connections and/or common public-key exchange methods.

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

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

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 (such as proven by Google/Apple's *Exposure Notification* framework [44]). 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 transmits it over the protocol's broadcast channel. *Bob* receives the packet and performs a lookup for TK_{A_i}. Upon successful match, *Bob* determines they are in fact the packet's intended destination and it was generated by *Alice*, retrieving the TK_{A_i}-paired REK_{A_i} to decrypt M_E.

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

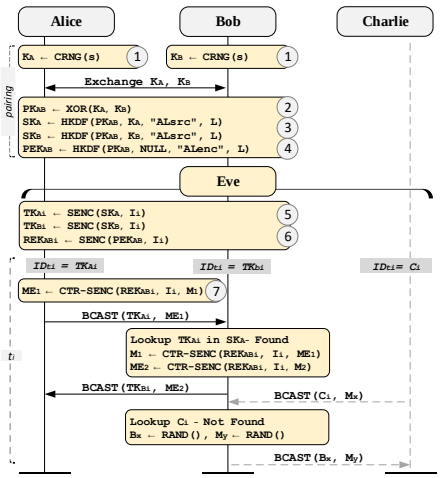

Fig. 7: Passive Deanonymization AL Mitigation

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.

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

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

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 as a pseudo-response. 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.

VI. Mitigation Evaluation

We implemented both Cache and Hash AL methods as a C library and evaluated overhead impact using on-device protocol simulations. The simulation baseline is an application protocol which generates data, incurs one megabit/sec transmission latency (simulating BLE 1M Phy/1Mbps), and processes the data upon receipt that varies in time with data size. We measured overhead impact from cryptographic and lookup operations relative to baseline during operation over varying sized packets. We executed each experiment ten times, removing the lowest and highest values, and averaging the remaining eight data points gathered from execution on a laptop PC with Intel Core i7-12700H 14 core processor and 32 GB of RAM running Ubuntu 22.04.3 and a Google Pixel 7 smartphone running Android 14. Additionally, we measure power consumption of the key resolution, encryption, and pseudo-response impact relative to baseline on both devices.

Size	Enc. (ms)	Dec. (ms)	Total (ms)	Base (ms)	Δ
Pixel 7 Smartphone					
16	0.00487	0.00506	0.00993	0.54841	1.81%
32	0.00551	0.00551	0.01102	0.68064	1.62%
64	0.00717	0.00736	0.01453	0.94310	1.54%
128	0.01074	0.01085	0.02159	1.46606	1.47%
256	0.01805	0.01852	0.03657	2.50955	1.46%
512	0.03887	0.03993	0.07880	4.62103	1.71%
1024	0.09839	0.10140	0.19979	8.86632	2.25%
2048	0.22943	0.24132	0.47076	17.33921	2.71%
PC Laptop					
16	0.00330	0.00483	0.00814	0.49123	1.66%
32	0.00385	0.00622	0.01007	0.62184	1.62%
64	0.00543	0.00934	0.01477	0.88302	1.67%
128	0.00870	0.01576	0.02446	1.40463	1.74%
256	0.01495	0.02752	0.04247	2.44767	1.74%
512	0.02619	0.04943	0.07562	4.53251	1.67%
1024	0.04908	0.09395	0.14304	8.70166	1.64%
2048	0.08932	0.17213	0.26145	17.03093	1.54%

TABLE VI: Average encryption overhead per packet

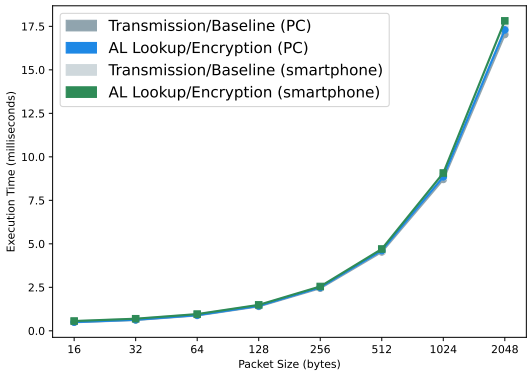

Fig. 8: Encryption & pair lookup overhead for Pixel 7 smartphone and PC laptop by packet size paired with a single device

Our source code for AL is available as a public GitHub repository at https://github.com/OSUSecLab/AnonymizationLayer.

Encryption. Table VI shows mean per-packet encryption overhead for sending and receiving data with sizes doubling from 16 to 2048 bytes (B) to account for a wide variety of protocol maximum transmission unit sizes when considering impact to effective throughput rate. Measuring across 10,000 packets, the mean overhead encryption per-packet across all data sizes is 1.82% on the smartphone, and 1.66% on the PC. The overhead is observed to be approximately linear relative to the data size, as illustrated in Figure 8.

	Hash (ms)			Cache (ms)		
Pairs	**Send**	**Recv.**	**Δ**	**Send**	**Recv.**	**Δ**
Pixel 7 Smartphone						
1	0.00117	0.02700	1.12%	0.00109	0.01742	0.74%
4	0.00139	0.05475	2.24%	0.00126	0.01842	0.78%
16	0.00188	0.17209	6.93%	0.00144	0.02014	0.86%
128	0.02821	1.32464	53.91%	0.00425	0.04013	1.77%
512	0.06770	5.57161	224.71%	0.03129	0.15183	7.30%
PC Laptop						
1	0.00099	0.01677	0.73%	0.00095	0.00619	0.29%
4	0.00100	0.04178	1.75%	0.00100	0.00606	0.29%
16	0.00114	0.11065	4.57%	0.00121	0.00645	0.31%
128	0.00361	0.39888	16.44%	0.00358	0.01080	0.59%
512	0.00975	1.33669	55.01%	0.01050	0.02280	1.36%

TABLE VII: Key resolution overhead for packet of varied sizes

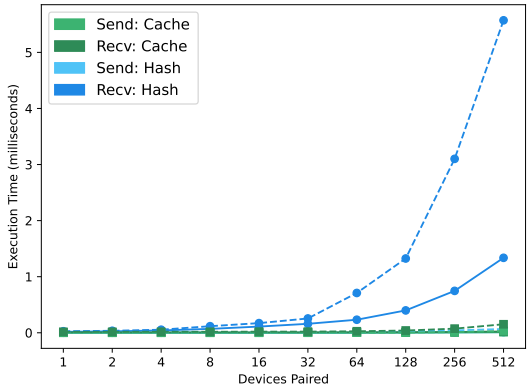

Fig. 9: Transmission key lookup and resolution overhead. PC in solid lines (—) and smartphone in dashed lines (- -)

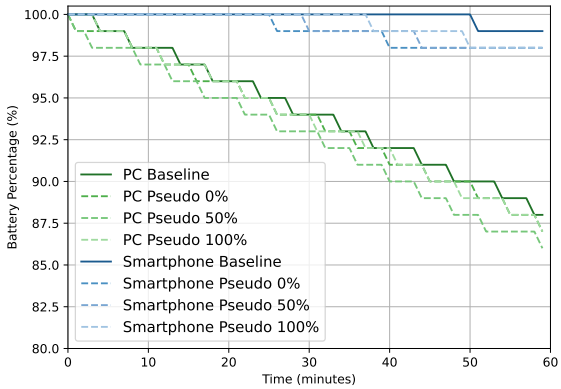

Fig. 10: Battery consumption for PC and smartphone over one hour, sending/receiving 20,000 packets/min, varying pseudo-response rates

Key Resolution. Table VII shows the summary worst-case key lookup and resolution overhead for send and receive functions of both Cache and Hash methods as device pairs doubles from 1 to 512 measured across 1 million total packets, with overhead percentages relative to a 256B packet baseline. Notably, our Cache method introduces approximately 0.02 ms (< 0.86%) and 0.18 ms (7.3%) overhead for up to 16 and 512 paired devices, respectively on the smartphone. While most smartphones may only have lower pair numbers, it demonstrates the performance increase over existing methods outperforming the Hash method by over 8x (16 pairs) and 30x (512 pairs). The PC performance of Cache is even greater, as expected, with over a 40x improvement over Hash, illustrated in Figure 9.

Power. As illustrated in Figure 10, the battery consumption overhead for PC and Pixel 7 smartphone baseline is 12% and 1%, respectively. Sending and receiving 20,000 packets (high traffic) per minute at varying pseudo-response rates of 0%, 50%, and 100%, we observe a 1-2% increase in power consumption for both devices over one hour. Notably, the pseudo-response rate does not have a significant impact on

battery consumption at the measured integer precision.

Evaluation Summary. Our evaluation shows AL as a viable IDBLEED countermeasure and real-world privacy-enhancing technology. Using our Cache method to support G_1 introduces less than 0.022 ms (0.86%) on the smartphone and 0.008 ms (0.31%) on the PC mean overhead for 16 device pairs. Encryption (G_2) introduces 0.037 ms (1.46%) and 0.042 ms (1.74%) on 256B packet sizes, on smartphone and PC respectively. These equate to approximately 2% communication overhead on both devices. Power consumption overhead is negligible at 1-2% per hour across tested devices with varying pseudo-response rates.

VII. DISCUSSION

IDBLEED Practicality. As first mentioned in threat model (§III-C), we assume an attacker can setup relays and the targeted devices are exclusive-use. With a known residence or location, and therefore proximity to their IoT peripheral devices, this becomes a practical exercise for a motivated attacker with standard sniffing equipment. Note, the attacker does not need to be present for initial pairing/bonding.

Alternatives to user tracking exist but come with technical challenges absent in IDBLEED. For example, using cameras to determine *Alice's* presence requires high bandwidth and power, along with either advanced computer vision models or human oversight to identify individuals in video feeds. This also sacrifices the programmatic packet processing and real-time notifications that IDBLEED provides. Optical methods have additional issues, such as requiring unobstructed views, visual discrepancies, and quality loss over distance, leading to unknown variability and loss of confidence. In contrast, IDBLEED enables real-time, programmatic de-anonymization with approximately and minimally *1/n* confidence, where *n* is the number of paired devices in an exclusive-use relationship. We find exclusive-use is common among IoT devices. Even if a device has multi-user support, it is often shared by small, associated groups, such as family members. In this scenario, IDBLEED can track family member locations, but could not differentiate between them. However, if not solely, it can contribute as a data point in a multi-factor approach to device deanonymization, thanks to its precision as a single-bit boolean that indicates a device's presence with an extremely low false-positive rate due to cryptographic and security mechanisms from the vulnerable protocols that produce the observable success traffic pattern.

The IDBLEED active attack offers the option of either relaying or replaying traffic, depending on the specifics of the protocol. However, we argue that replay attacks are generally more practical than relay, which requires that both targeted devices process packets at the same time. It would be problematic if a device disables its wireless interface in sleep mode to conserve power, as is often the case with certain wireless keyboards and a small percentage of other devices. Our research indicates that many stages of wireless protocols remain susceptible to replay attacks. For example, BLE devices that use an *LTK* to derive a *SK* for encrypting packets. Although 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`—lack these protective measures against replay attacks.

IDBLEED Limitations. Examining the details and limitations of IDBLEED allows further understanding of how the vulnerabilities have evaded previous research. First, as mentioned in §III-C, a device must have at least one exclusive-use, trusted relationship — devices without this are not susceptible to IDBLEED without other ways to identify success/failure communication patterns. These limitations imply a level of apriori knowledge of the victim — however, remains feasible for a motivated attacker and certainly for nation-states. Indeed, IDBLEED attacks do not directly identify users of an arbitrary device, discovering their personally identifiable information. Therefore, IDBLEED is a targeted attack and does not generally scale to deanonymize arbitrary or mass populations of devices.

Second, our paper primarily demonstrates several IDBLEED attack scenarios requiring close proximity (e.g., 10-30 meters in indoor environments for BLE [4]), with the aim of illustrating the wireless communication component of IoT. However, the exclusive-use attack principle applies in broader contexts, even devices controlled remotely via HTTPS, which may exhibit an exclusive-use pattern. Meanwhile, IDBLEED comes at a lower cost, stealth, and processing power than other remote tracking methods that utilize other mediums such as video or audio detection. IDBLEED effectively sniffs, dissects, and relays packets compared to recording video, audio, which requires high on-device storage and/or processing using detection algorithms. These methods are limited similarly by proximity, but additionally must consider obstruction of view or audibly noisy environments, and data transmission bandwidth or node/edge-device computation.

Third, we recognize target devices may need to be in a specific protocol state for successful IDBLEED attacks. For example, when `LL_ENC_REQ` is relayed from a trusted smartphone to a BLE peripheral, the two devices are already connected and in the communication stage. Despite this, the attacker can capture and retransmit specific packets between the devices or create packets that follow the protocol specification to restart the connection process and begin the attack again.

IDBLEED Generality. Our primary focus, detailed in §III, is on BLE and Wi-Fi. However, our findings show that IDBLEED's core concept of deanonymization applies to any protocol with exclusive-use characteristics that leak a boolean traffic pattern side-channel. Motivated by this, we explored various IoT devices and companion apps at the user application layer to demonstrate broader and real-world applicability, provided in §A for additional reading.

IDBLEED Impact & Consequences. IDBLEED introduces a new form of tracking attack, exploiting a specification flaw

in widely-used protocols that leak a boolean side-channel. Unlike vulnerabilities based on implementation flaws, the IDBLEED exploitation window is prolonged, as specification flaws are less likely to receive quick fixes. Tracking attacks like IDBLEED pose significant privacy and safety risks, diminishing a person's freedom of movement and enabling threats such as human trafficking, stalking, or pattern-of-life analysis, especially for high-profile individuals. Knowing a person's location can be the starting point for more severe digital or physical threats.

Anonymization Layer Practicality. An approximate 2% mean increase measured on-device for both power consumption and single-trip packet transmission/processing is negligible and greatly supports AL's practicality for point-to-point communication anonymization. Additional optimizations can be done, for example, both Cache and Hash anonymity methods can store recently communicated pair records to decrease lookup times, similar to Most Recently Used (MRU) in memory management. Least Recently Used (LRU) or similar methods can be utilized to evict source transmission keys from pair records as a more dynamic, non-time based key management mechanism. Encryption performance can be increased through algorithm optimization or dedicated hardware. Additional investigation is required into scalability in more dynamic and autonomous IoT systems that may interact 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 as an anonymized geo-fenced initial hash table lookup may decrease the number of keys to check using Hash or Cache methods.

Anonymization Layer Security Analysis. AL design goals (§V) were born by adopting an adversarial mindset and ultimately aim to reduce the amount of meta and side-channel information available via packet inspection and to remove the observable boolean exclusive-use side-channel that we've introduced as the core foundation of IDBLEED. We accomplish this by removing transmission directionality to remove association and precise data flow analysis; 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-response packets indistinguishable from those sent to trusted entities given the randomly rotating, ephemeral source identifiers and entropy in payload. The brute-force success probability is $\frac{N}{2^{48}}$ given N number of 6 byte transmission keys, and $\frac{1}{2^{256}}$ for 256-bit encryption keys.

A mitigation analysis for active IDBLEED attacks is illustrated in Figure 11, omitting the workflow already introduced in Figure 7 for brevity. At time t_i, *Eve* captures a packet at $LocationA$ from anonymous *Alice* with identifier X and relays it to *Bob*. 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

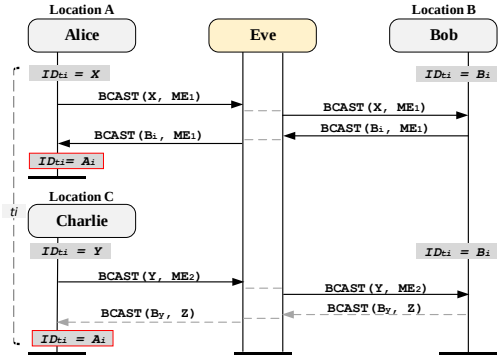

Fig. 11: Active Deanonymization Mitigated by AL

random bytes as pseudo-communication. Given the exclusive-use characteristic, *Eve* must believe *Alice* is at both locations, a logical contradiction, and therefore concludes *Alice*'s location can not be accurately or confidently determined.

Anonymization Layer Limitations. Our defense is not perfect. Certain hardware throughput limitations may degrade pseudo-responses effectiveness in order to maintain timing and quality of service requirements, possibly rendering devices vulnerable to statistical analysis, particularly in attacker-controlled packet flooding. We discuss two scenarios based on the number of nearby devices: (i), the attacker is near many devices broadcasting with anonymized source addresses creating a noisy environment, complicating identification of legitimate responses from a victim device. (ii), the attacker operates in a controlled environment where only the target device is present, enabling precise statistical evaluation during packet flooding. However, this scenario is beyond the scope of anonymous communication solutions — if the attacker has controlled the environment well enough to know the victim device's presence, determining its location is unnecessary because it is already apparently known. Between these scenarios lies a spectrum of device numbers, where results would require advanced statistical analysis, leading to ambiguous confidence metrics influenced by various factors. Ultimately, the anonymization layer discourages reliance on this method, prompting attackers to likely seek more reliable but higher-cost tracking methods.

Anonymization Layer Generality. While AL was designed to mitigate IDBLEED for IoT, we recognize its benefits generalize to provide anonymity at any layer or point of integration in a communication stack. This concept extends even more abstractly to generalized information exchange, provided there is a broadcast-like capability for the communication medium.

Ethics and Responsible Disclosure. We adhered to the highest ethical standards performing our experiments, which were conducted in a controlled laboratory environment without targeting external devices. We disclosed technical details of the BLE and Wi-Fi attacks, along with potential countermeasures, to the tested device manufacturers and the Bluetooth Special Interest Group (SIG). Several companies acknowledged our findings, successfully replicated the attack,

and plan further investigations.

VIII. Related Works

IoT Privacy Attacks. Previous efforts have discovered various ways IoT devices can be tracked through their wireless communication. These include using unique identifiers such as MAC addresses and IP addresses, as well as analysis of side-channel information such as network traffic and power usage patterns. For example, leaked information in encrypted traffic, such as network connection frequency and a device's DNS server connection, can be used to fingerprint IoT devices [14], or machine learning algorithms analyzing network traces to identify IoT devices [15], [16], [17]. Huang et al. [18] demonstrated an attacker can infer user behavior and lifestyle based on IoT device state changes and associated network traffic.

Numerous Bluetooth device tracking attacks have been proposed, including those using sniffers to collect advertising packets [19], [20], [21], [6], [2], [45], [22], [23], [46], [3], [24], [5], [25], [13]. Previous approaches such as *BlueTrack* [3] and *BLEB* [24] 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 [21], [6], [26], [27] and wearable fitness trackers [47]. Some attacks rely on static payloads to track devices, such as manufacturer identifiers [2], [20], information elements [28], [29], and GATT attributes [48]. Serhan et al. [49] introduce a real-time probabilistic framework for tracking BLE devices using a hidden Markov model, which uses data-driven probabilistic radio-frequency maps from received signal strength indicators (RSSI). *BLENDER* [50] framework fingerprints BLE devices by analyzing standard BLE interactions using machine learning to predict device presence based on time and location. Jianliang et al. [51] present a formal model for assessing BLE untraceability, revealing vulnerabilities that allow user tracking despite MAC address randomization. Tingfeng et al. [52] examine security and privacy aspects of Samsung's Offline Finding (OF) protocol used to locate BLE devices, including the potential for tracking OF devices, tracking of individuals, and location de-anonymization.

Tracking attacks have also been developed for Wi-Fi networks. Sapiezynski et al. [53] 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. [30] developed a passive tracking system, *Probr*, that manages various types of Wi-Fi capture devices and processes collected traces. Petre et al. [31] demonstrated the effectiveness of Wi-Fi tracking at large events exceeding 100,000 people over three days. *RF-Track* [54] is an indoor location attack on Wi-Fi devices, using a Reinforcement Learning agent to analyze RSSI sequences and build a fingerprint map. *CSI-RFF* [55] uses micro-signals within Channel State Information to fingerprint Wi-Fi devices.

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

TABLE VIII: Comparison between IDBleed and *BAT* attacks.

Our IDBleed attack differs significantly from previous studies for three reasons: First, it exploits patterns observable from the trusted relationship between devices, rather than relying on static patterns in network traffic ([21], [6], [26], [27]) or power usage. Second, it can deanonymize and track devices by sending a single packet, unlike previous research requiring large volumes of packet collection. Third, attackers can actively probe devices to gather data instead of passively waiting for packets from inactive devices.

While *BAT* attacks [7] were inspirational, our study differs significantly from theirs in the following ways. Highlighted in Table VIII, *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, their study relies on 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. Finally, our boolean 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 is an entirely new communication stack layer that enhances privacy.

Anonymization Defense. Various anonymous communication solutions have been proposed for IoT networks. Palmieri et al. [56] 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 [44] uses random rotating identifiers with payloads that are broadcasted and captured by nearby participating devices, with other works proposing privacy-enhancing modifications to safeguard against relay and replay attacks [57], [58]. *BLE-Guardian* [20] allows a user to customize the accessibility of advertising packets. Hadi Givehchian et al. present [59], a method to obfuscate physical-layer fingerprints in BLE devices, which attackers exploit to bypass MAC address randomization and track devices. Tor [60] uses onion routing for primarily TCP-based applications to provide an anonymous communication service consisting of operator-maintained nodes. However, these solutions require support from intermediate servers or gateway

nodes to provide their privacy preserving functions. In contrast, our proposed AL exchanges keys and resolves communication directly on devices without requiring additional infrastructure or third-party configuration. Zhang et al. propose *MASK* for mobile ad-hoc networks for anonymized single-hop communication [61] 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, these solutions are different from ours in the following aspects. First, many require an explicit authentication and key exchange prior to a new session of data communication. Consequently, they are susceptible to IDBLEED, as this exclusive-use three-way handshake authentication mechanism is observable to either succeed or fail given subsequent communication. Our solution with AL is transparent to other layers — there is no modification to existing layers and they retain established roles, 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. Second, even solutions without additional key exchanges ([20], [59]) still require the defender, which may be a centralized device, to monitor the traffic to identify potentially vulnerable or malicious devices. This approach can introduce additional costs associated with continuous monitoring.

IX. Conclusion

We have shown ubiquitous wireless communication protocols, such as BLE and Wi-Fi, are vulnerable to deanonymization and tracking attacks, despite modern countermeasures such as address randomization. The trusted association between paired devices produces differences in traffic patterns that leak an observable boolean side-channel, a historically overlooked scenario we have termed *exclusive-use*. This fundamental flaw and vulnerability used to deanonymize devices is the key component to our practical IDBLEED tracking attacks. We propose a generalized mitigation featuring our ANONYMIZATION LAYER and observe its on-device performance impact to be negligible as a viable privacy-enhancing technology.

Acknowledgment

We thank reviewers for their insightful comments, which have significantly improved the paper. This work was partially supported by National Security Agency under contract H98230-23-C-0282, and National Science Foundation through grants CNS-2112471 and CNS-2207202.

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] C. Lei, Z. Ling, Y. Zhang, Y. Yang, J. Luo, and X. Fu, "A friend's eye is a good mirror: Synthesizing {MCU} peripheral models from peripheral drivers," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 7085–7102.

[10] K. Liu, M. Yang, Z. Ling, Y. Zhang, C. Lei, J. Luo, and X. Fu, "Riotfuzzer: Companion app assisted remote fuzzing for detecting vulnerabilities in iot devices," in *Proceedings of the 31th Conference on Computer and Communications Security (CCS'24)*, 2024.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] ——, "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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] M. K. Jangid, Y. Zhang, and Z. Lin, "Extrapolating formal analysis to uncover attacks in bluetooth passkey entry pairing." in *NDSS*, 2023.

[34] 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.

[35] 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

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

[37] 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.

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

[39] 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.

[40] "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.

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

[42] 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

[43] 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.

[44] "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).

[45] Y. Zhang, J. Weng, Z. Ling, B. Pearson, and X. Fu, "Bless: A ble application security scanning framework," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 636–645.

[46] C. Zuo, H. Wen, Z. Lin, and Y. Zhang, "Automatic fingerprinting of vulnerable ble iot devices with static uuids from mobile apps," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1469–1483.

[47] 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.

[48] 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.

[49] F. S. Daniş, C. Ersoy, and A. T. Cemgil, "Probabilistic indoor tracking of bluetooth low-energy beacons," *Performance Evaluation*, vol. 162, p. 102374, 2023.

[50] K. Szyc, M. Nikodem, and M. Zdunek, "Bluetooth low energy indoor localization for large industrial areas and limited infrastructure," *Ad Hoc Networks*, vol. 139, p. 103024, 2023.

[51] J. Wu, P. Traynor, D. Xu, D. J. Tian, and A. Bianchi, "Finding traceability attacks in the bluetooth low energy specification and its implementations," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 4499–4516. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/wu-jianliang

[52] T. Yu, J. Henderson, A. Tiu, and T. Haines, "Security and privacy analysis of samsung's Crowd-Sourced bluetooth location tracking system," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 5449–5466. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/yu-tingfeng

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

[54] R. Li, H. Hu, and Q. Ye, "Rftrack: Stealthy location inference and tracking attack on wi-fi devices," *IEEE Transactions on Information Forensics and Security*, 2024.

[55] R. Kong and H. Chen, "Csi-rff: Leveraging micro-signals on csi for rf fingerprinting of commodity wifi," *IEEE Transactions on Information Forensics and Security*, 2024.

[56] "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).

[57] C. Ellis, H. Wen, Z. Lin, and A. Arora, "Replay (far) away: Exploiting and fixing google/apple exposure notification contact tracing," *Proceedings on Privacy Enhancing Technologies*, vol. 2022, pp. 727–745, 10 2022.

[58] M. Casagrande, M. Conti, and E. Losiouk, "Contact tracing made un-relay-able," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 221–232. [Online]. Available: https://doi.org/10.1145/3422337.3447829

[59] H. Givehchian, N. Bhaskar, A. Redding, H. Zhao, A. Schulman, and D. Bharadia, "Practical obfuscation of ble physical-layer fingerprints on mobile devices," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 2867–2885.

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

[61] 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

A. IDBLEED *Attacks on User Applications*

As additional reading, we present a sampling of six companion smartphone apps for IoT devices found to be vulnerable to IDBLEED. These include commonly used household or smarthome IoT devices such as blood pressure monitors, smartlocks, smartplugs, and smartlights. Our reverse

engineering and manual analysis discovers that the exclusive-use characteristic is present on all tested devices, leaving their trusted smartphones vulnerable to deanonymization and tracking attacks through IDBLEED, as summarized in Table IX. Each case study introduces a high-level protocol and attack workflow.

Device	Type	Channel	Exclusive-Use	Passive Attacks	Active Attacks
AppLights Standards	Light	BLE	❸ ❹	✓	✓
AppLights C9	Light	BLE	❶ ❸ ❹	✓	✓
AppLights Strings	Light	BLE	❶ ❸ ❹	✓	✓
i-Health Labs	Medical	BLE	❸ ❹	✓	✓
Ultraloq	Lock	BLE	❶ ❷ ❸ ❹	✓	✓
Kasa Plug	Plug	Wi-Fi	❸ ❹	✓	✓

TABLE IX: Tested IoT devices. ❶ Verification, ❷ Encryption, ❸ Authentication, ❹ Auto-connection

IDBLEED in Blood Pressure Monitors. Our investigation of blood pressure monitors from i-Health Labs first reveals they broadcast basic information to nearby devices over BLE, including manufacturer and device name. The smartphone companion app receives these packets and verifies if the device is recognized on its auto-connection list of trusted devices. If so, the app and monitor device authenticate using a challenge-response protocol with a fixed key and begin communicating using encryption if successful.

Evaluation. The exclusive-use characteristic is present at the trusted device verification stage of authentication, leaving the smartphone vulnerable to both passive and active IDBLEED attacks. Because a fixed key is used, we find this communication method is also vulnerable to the active replay variant. Further, we discover the encryption key is based on the device specific hardware version number, and therefore if unchanged, leaves any post-authentication communication of measurements also vulnerable to replay. We also find the companion app receives BLE beacons even in the background. These findings widen susceptibility for deanonymization and allow data poisoning of critical health information. The root cause of the vulnerability is the presence of a trusted list of blood monitor static MAC addresses stored by the companion app that associates the two devices, allowing *Eve* to spoof a previously observed trusted address.

IDBLEED in Smartlocks. For our next case study, we investigate Ultraloq smartlocks and find a secure workflow to ensure secure communication between the lock and associated companion app. This process begins with an initial setup that requires a password to be set, stored on the smartphone, and transmitted to share with the smartlock. For subsequent connections, the smartlock initiates a challenge-response protocol to authenticate the smartphone. The smartlock sends a random key, which the companion app encrypts with a control command using the pre-shared password, and sends it back to the smartlock. The smartlock decrypts the message and compares the value to its original random key to determine authenticity.

Evaluation. We find the Ultraloq smartlock to be vulnerable to both passive and active IDBLEED attacks during the authentication stage when the smartphone replies to the smartlock request with an encrypted message. The assumption is that *Eve* is not present during the initial setup process and observes the password. However, while the attack window to capture the initial pairing is small, the active relay variant is still possible.

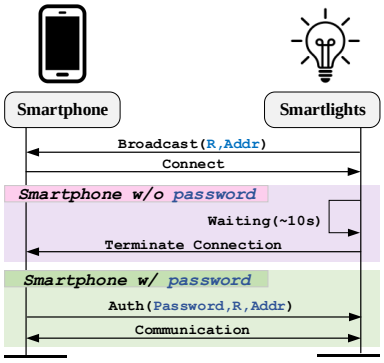

Fig. 12: AppLights C9 & Strings smartlights workflow

IDBLEED in Smartplugs. Investigating Kasa smartplugs, we find the companion app receives IPv4 packets and uses the address to determine if the device is trusted before initiating a connection. The smartplug initially authenticates the app using credentials pre-shared from the initial pairing process, however, the credentials are not used in subsequent sessions. Instead, the companion app uses an encoding function that takes the key as input to encode commands to transmit to the smartplug.

Evaluation. Once again, we discover the Kasa smartplugs are vulnerable to both passive and active IDBLEED attacks. This is due to the static IPv4 address being saved by the companion app after initial pairing. Therefore, *Eve* can passively observe exchanges with the IPv4 address if on the same network as the two devices, or spoof the address if on the same network as the smartphone to deanonymize *Alice*.

IDBLEED in Smartlights. Our final case study investigates three versions of smartlights from AppLights: Standards, C9, and Strings. All versions broadcast advertisements that include its BLE address. With Standards, the companion app receives the advertisement and simply checks for the address on its auto-connection list, sending a connection request if it exists. Once connected, the companion app sends authentication data to the smartlights, generated using the pre-shared password, the smartlight's static BLE address, and the smartlight's static firmware information.

The workflow of the C9 and Strings versions are similar, as shown in Figure 12, which accurately reflects the real-world traces in Table X. Building off the Standards version, their advertisements also contain a randomly generated byte array in addition to the static BLE address to add a challenge-response mechanism. The companion app includes a random integer from the received array in its response. The smartlights verify the integer chosen by the companion app is indeed in its original array and accepts commands if so.

No.	Time	Source ID	Destination ID	PDU Type	PDU Playload
Smartphone w/o Password					
1	00:36	98:7b:f3:78:d0:ad	Broadcast	ADV_IND	ADDR
2	00:40	ad:d8:3e:a9:ba:52	98:7b:f3:78:d0:ad	SCAN_REQ	EMPTY
3	00:44	98:7b:f3:78:d0:ad	Broadcast	SCAN_RSP	RAND
4	00:48	ad:d8:3e:a9:ba:52	98:7b:f3:78:d0:ad	CONNECT_REQ	EMPTY
5	01:00	ad:d8:3e:a9:ba:52	98:7b:f3:78:d0:ad	ATT_WRITE	DISCOVER_SERVICES
6	01:04	98:7b:f3:78:d0:ad	ad:d8:3e:a9:ba:52	ATT_READ	0xFFF1
7	01:12	ad:d8:3e:a9:ba:52	98:7b:f3:78:d0:ad	EMPTY_PDU	EMPTY
8	01:22	98:7b:f3:78:d0:ad	ad:d8:3e:a9:ba:52	LL_TERMINATE_IND	EMPTY
Smartphone w/ Password					
1	00:36	98:7b:f3:78:d0:ad	Broadcast	ADV_IND	ADDR
2	00:40	ad:d8:3e:a9:ba:52	98:7b:f3:78:d0:ad	SCAN_REQ	EMPTY
3	00:44	98:7b:f3:78:d0:ad	Broadcast	SCAN_RSP	RAND
4	00:48	ad:d8:3e:a9:ba:52	98:7b:f3:78:d0:ad	CONNECT_REQ	EMPTY
5	01:00	ad:d8:3e:a9:ba:52	98:7b:f3:78:d0:ad	ATT_WRITE	DISCOVER_SERVICES
6	01:04	98:7b:f3:78:d0:ad	ad:d8:3e:a9:ba:52	ATT_READ	0xFFF1
7	01:12	ad:d8:3e:a9:ba:52	98:7b:f3:78:d0:ad	ATT_WRITE	AUTH(PASS, RAND, ADDR)
8	01:16	98:7b:f3:78:d0:ad	ad:d8:3e:a9:ba:52	ATT_READ	NOTIFY(OK)

TABLE X: Excerpt of AppLights packet traces.

Evaluation. While all three versions of the lights are susceptible to passive and active IDBLEED attacks, we observe the addition of the challenge-response mechanism in the C9 and Strings versions is a defense against the replay variant. However, the Standards version is vulnerable to replay, as the authentication packet is the same every time. As a recurring theme in companion apps, the deanonymization occurs when the companion app verifies and responds to the trusted device after it simply checks for the static MAC address in its auto-connection list.