SoK: The Long Journey of Exploiting and Defending the Legacy of King Harald Bluetooth

Jianliang Wu
Purdue University & Simon Fraser University
wu1220@purdue.edu

Ruoyu Wu
Purdue University
wu1377@purdue.edu

Dongyan Xu
Purdue University
dxu@purdue.edu

Dave (Jing) Tian
Purdue University
daveti@purdue.edu

Antonio Bianchi
Purdue University
antoniob@purdue.edu

***Abstract*—Named after the Viking King Harald Bluetooth, Bluetooth is the de facto standard for short-range wireless communications. The introduction of Bluetooth Low Energy (BLE) and Mesh protocols has further paved the way for its domination in the era of IoT and 5G. Meanwhile, attacks against Bluetooth, such as BlueBorne, BleedingBit, KNOB, BIAS, and BLESA, have been booming in the past few years, impacting billions of devices. While Bluetooth security has drawn significant attention from the security research community, a systematic understanding of this field is still missing, impeding the advancement of this field.**

In this paper, we first summarize the evolution of Bluetooth security in the specification in the past 24 years. Then, we provide a systematization of Bluetooth security by diving into 76 attacks and 33 defenses presented by previous research in this area. We first categorize attacks and defenses based on their affected layers and protocols in the Bluetooth stack as well as their threat models. Then, we cross-check the attacks and defenses to have a big picture of Bluetooth security. Based on the systematization, we find that the existing formal analyses of Bluetooth do not cover most of the security aspects of Bluetooth Mesh. Lastly, we take a step towards securing Bluetooth Mesh by designing and implementing a comprehensive formal model of Bluetooth Mesh covering all its security-related protocols. Our systematization reveals, for instance, that the security of Bluetooth pairing faces challenges caused by users' mistakes, and that Bluetooth fuzzing is effective yet not comprehensive. Based on the systematization, we provide promising future directions to shed some light on future Bluetooth security research.

1. Introduction

After the Viking King Harald Bluetooth [1] united the Danish tribes over 1,000 years ago, his "legacy", the Bluetooth protocol, has become the de facto standard for short-range wireless communications today. Since the introduction of Bluetooth (Basic Rate, BR) in 1999, new features, such as Enhanced Data Rate (EDR) and High Speed (HS), have been added to improve its performance. Later, Bluetooth Low Energy (BLE) and Bluetooth Mesh (Mesh) were released paving the way for its domination in the era of Internet of Things (IoT) and 5G. Together with Bluetooth Classic (BR/EDR/HS), the Bluetooth protocol suite powers billions of devices [2].

Meanwhile, attacks against Bluetooth (e.g., BlueBorne [3], BleedingBit [4], KNOB [5], [6], BIAS [7], and BLESA [8]) have been booming in the past few years, impacting billions of devices. These attacks exploit both design issues in the Bluetooth specification and implementation flaws in its implementations allowing for privilege escalation, remote code execution, breaking cryptography, spoofing, etc. Correspondingly, different defense solutions have been proposed to address these flaws within Bluetooth, including Linux eBPF modules [9], InternalBlue [10], Frankenstein [11], and BlueShield [12]. These defenses enable Bluetooth packet filtering, firmware analysis, firmware fuzzing, device identification, etc., by leveraging different infrastructures, frameworks, and physical characteristics.

While Bluetooth security as a research field has already drawn significant attention from the community, a systematic understanding of this field is still missing, impeding the advancement of this field. For example, no existing literature presents how the Bluetooth security in the specification evolved during the past 24 years and what the status quo is. Moreover, due to its complexity, it is not well-explored whether the attack model in the specification aligns with the one in real-world attacks and defenses. In terms of attacks, it is still unclear what the root causes are of existing attacks, how these attacks relate to each other and relate to different parts of the protocol, and whether they target only one Bluetooth protocol, e.g., Bluetooth Classic (BC), or apply to other protocols. Similarly, on the defense side, it is still unclear what the most effective and/or least intrusive defenses might be, how these defenses relate to each other and target different parts of the protocol, and whether existing defenses can reasonably cover known attacks.

To address these issues, *in this paper, we provide a systematic study to categorize what has been done in the field of Bluetooth security, and to highlight what is missing and should be done as future work.*

We first provide an overview of the evolution of Bluetooth security and privacy features since its first version. Then, we systematize the field of Bluetooth security by diving into 76 attacks and 33 defenses. To reason about known attacks, we categorize them based on the layers of the Bluetooth stack they target and the attack techniques they use. Similarly, we categorize defenses according to the layers of the Bluetooth stack they are designed to protect and the defense techniques they use. To provide more insights, we also specify their scope (e.g., targeting only BC or more), affected phase (e.g., device discovery phase and data transmission phase), and attack models (e.g., whether the device is compromised or not). For defenses, we also study the prerequisites (i.e., hardware and software requirements) of each defense. These studies allow us to cross-check the attack and defense systematization to have a big picture of today's Bluetooth security. Our findings, to name a few, include: 1) the BLE device discovery phase draws most privacy attacks; 2) pairing security faces challenges caused by users' mistakes; 3) Bluetooth specification has different assumptions from its implementations on modern operating systems (OSes); 4) while being effective in finding firmware vulnerabilities, Bluetooth fuzzing has limited support for the host code; 5) there is no effective defense against attacks exploiting users' mistakes; 6) defenses against attacks targeting BC and BLE encryption are still missing.

Building upon our systematization, we find that formal analysis of Bluetooth is effective [8], [13]–[15] in finding design vulnerabilities in the specification. However, the existing formal analysis of Mesh [13] only considers its provisioning protocol, leaving other security-related protocols (e.g., key refresh) uncovered. Thus, to fill this gap, we take one step towards securing Mesh by *proposing a comprehensive Mesh formal model covering all security-related protocols*. We implement our formal model using ProVerif [16] and verify 11 properties covering all 8 different modes. Using our model, we rediscover 2 known attacks against Mesh. In addition, we also use our model to confirm that Mesh is not vulnerable to any known attacks anymore after applying the fixes suggested by Bluetooth SIG. To foster future research in this field, our model is publicly available [17]. To the best of our knowledge, our work is the first to systematically examine Bluetooth security. Moreover, we provide promising directions, e.g., more comprehensive exploration of BLE and Mesh privacy, cross-stack security and privacy evaluation, and comprehensive Bluetooth fuzzing, to shed some light on future Bluetooth security research.

2. Bluetooth Evolution

Bluetooth is a short-range radio frequency standard, maintained by the Bluetooth Special Interest Group (SIG), for data exchange between different devices, such as smartphones, laptops, and headsets. It works at the 2.4 GHz industrial, scientific, and medical frequency bands. During the past 24 years, from version 1.0 to version 5.4 (the latest version at the time of the paper writing), Bluetooth has evolved into three different protocols, namely Bluetooth Classic, Bluetooth Low Energy, and Bluetooth Mesh.

Regardless of the used protocol, Bluetooth communication can be divided into three phases: *device discovery*, *key sharing*, and *data exchange*. In the following of this section, we first describe how each of the protocols works in these three phases, then the evolution of its security and privacy features, and lastly the status quo of each protocol's security and privacy. In Table 1, we summarize the main attacks exploiting the design weaknesses in Bluetooth security features, their affected Bluetooth versions, and corresponding mitigations.

2.1. Bluetooth Classic

Bluetooth Classic (BC), which is primarily used for audio streaming, includes three different modes: Basic Rate (introduced in Bluetooth 1.0 [38]), Enhanced Data Rate (introduced in Bluetooth 2.0 [39]), and High Speed (introduced in Bluetooth 3.0 [40] and removed since Bluetooth 5.3 [41]). BC works in a central-peripheral scheme to form a piconet [41, p.1405]. The initiator of the connection usually acts as the *central* (e.g., a smartphone), and the responder (e.g., a headset) acts as the *peripheral*.

To communicate, two devices first discover each other in the device discovery phase using an inquiry-response procedure. The central BC device sends inquiry requests, and the peripheral device responds with inquiry responses to be discovered by the central device. After that, BC leverages a *pairing* process, which may require the user's confirmation (e.g., pressing a button), for the two devices to agree on a shared secret key. The two devices can store this key for future use. Lastly, in the data exchange phase, two BC devices first conduct a challenge-response scheme authentication procedure to verify they share the same secret key. Then, they use the shared secret key to encrypt/decrypt the exchanged data.

2.1.1. Security Evolution.

The security mechanism of BC involves three aspects: pairing, authentication, and encryption. We detail each of the three aspects.

Pairing. In the early versions (Bluetooth 1.x and 2.0), BC uses a legacy pairing protocol, during which the user needs to input the same Personal Identification Number (PIN) on both devices. In the legacy pairing protocol, the four-digit PIN [41, p.268] is the only source of entropy for a passive attacker to derive the shared secret key. As such, the attacker can sniff the traffic during pairing and brute-force the PIN value to recover the shared secret key [18]–[20]. In addition, if a device supports connecting to another device with the same MAC address as itself, an attacker can launch reflection attacks [21] to get the secret key even without knowing the PIN.

To address the above weaknesses, Secure Simple Pairing (SSP) was introduced to replace the legacy pairing protocol in Bluetooth 2.1 [42]. SSP adopts the Diffie-Hellman Key Exchange (DHKE) protocol to defend against passive attacks and introduces four pairing methods, namely Just Works (JW), Out of Band (OOB), Passkey Entry (PE), and Numeric Comparison (NC). The method is chosen based on the user I/O interfaces (e.g., screen and keyboard) of the devices. NC and PE are authenticated since user confirmation, such as pressing a button or inputting a PIN, is required during pairing. For OOB, whether it is authenticated relies on whether the OOB channel is authenticated or not. Since JW does not involve any user confirmation, it is unauthenticated and vulnerable to Man-in-the-Middle (MitM) attacks [43]–[48].

In the latter versions (3.x, 4.x, and 5.x), the SSP protocol does not change significantly except adding requirements to mitigate certain attacks. Since version 5.1, the passkey in PE should be randomly generated in each pairing [49, p.1087] to defend against the attacks exploiting fixed or reused passkeys [22]–[24]. Also since version 5.1, a device should validate the received key during DHKE to protect it from attacks exploiting invalid public keys [15], [25]. In the 5.3 version, the specification states that a device should fail the pairing process if the received

TABLE 1: Attacks exploiting design weaknesses in the specification, their affected Bluetooth versions and security features, and corresponding mitigations. Text in the pink bar : the security feature affected by an attack across Bluetooth versions.

Protocol	Mechanism	Attack	Bluetooth Version														Mitigation
			1.0	1.1	1.2	2.0	2.1	3.0	4.0	4.1	4.2	5.0	5.1	5.2	5.3	5.4	
BC	Pairing	PIN Brute-force [18]–[20]	BC Legacy Pairing	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	SSP
		Reflection Attack [21]	BC Legacy Pairing	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨			Spec. Update
		Passkey Reuse [22]–[24]					SSP	▨	▨	▨	▨	▨					Spec. Update
		Invalid Key Attack [15], [25]					SSP	▨	▨	▨	▨	▨					Spec. Update
		Reflection Attack [21]					SSP	▨	▨	▨	▨	▨	▨	▨			Spec. Update
	Authentication	Legacy Auth. Bypass [7]	Legacy Authentication	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	Advisory [26]
		Secure Auth. Bypass [7]								Secure Authentication	▨	▨	▨	▨			Spec. Update
	Encryption	Correlation Attack [27]–[31]	E0 Encryption	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	AES-CCM
		E0 Key Brute-force [5]	E0 Encryption	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				Spec. Update
		AES Key Brute-force [5]								AES-CCM	▨	▨	▨				Spec. Update
BLE	Pairing	Key recovery [32]							BLE Legacy Pairing	▨	▨	▨	▨	▨	▨	▨	SCP
		Reflection Attack [21]							BLE Legacy Pairing	▨	▨	▨	▨	▨			Spec. Update
		Passkey Reuse [22]–[24]									SCP	▨					Spec. Update
		Invalid Key Attack [15], [25]									SCP	▨					Spec. Update
		Reflection Attack [21]									SCP	▨	▨	▨			Spec. Update
	Encryption	AES Key Brute-force [6]							AES-CCM	▨	▨	▨	▨	▨	▨	▨	SCO Mode
		Device fingerprinting [33], [34]							Advertising	▨	▨	▨	▨	▨	▨		Spec. Update
Mesh	Configuration	PIN Brute-force [21]											Provisioning	▨	▨	▨	Advisory [35]
		Commitment Malleability [21]											Provisioning	▨	▨	▨	Advisory [36]
		Reflection Attack [13], [21]											Provisioning	▨	▨	▨	Advisory [37]

public key (or part of the key) is identical to its own [41, p.983] to defend against reflection attacks [21] against PE.

Authentication. From Bluetooth 1.0 to 4.0 [50], BC uses the *legacy authentication* procedure, which provides unilateral authentication. Though this procedure can be applied twice to achieve bilateral authentication [50, p.1114], the specification does not mandate bilateral authentication. Accordingly, an attacker can exploit the role switch feature (a feature that allows two devices to switch their central and peripheral roles [41, p.531]) to switch her role with the victim device's role. Via role switching, the attacker changes her role from the authenticatee (peripheral) to the authenticator (central) so that she can bypass the authentication on the victim device [7].

Since Bluetooth 4.1, the SIG has introduced the *secure authentication* procedure that mandates bilateral authentication. However, secure authentication can also be bypassed by switching the roles of two devices in the middle of the authentication [7]. Starting from version 5.3, the specification [41, p.973] forbids the role switch in the authentication procedure to mitigate the authentication bypass attack. Note that without the secret key, even though an attacker can bypass the authentication, she cannot communicate with the victim device if the encryption is properly performed.

Encryption. From Bluetooth 1.0 to 4.0, BC uses the E0 [51] stream cipher as the encryption algorithm for BC data exchange, which is vulnerable to correlation attacks [27]–[31]. To mitigate the attacks against E0, Bluetooth 4.1 [52] introduced the AES-CCM [53] algorithm for encryption. Since then, the encryption mechanism has remained unchanged except for adding additional requirements to mitigate certain attacks. In version 5.2 [54, p.631], the specification requires that the entropy of the encryption key is at least 7 bytes to mitigate the KNOB attacks [5] against encryption exploiting the allowance of and encryption key with one-byte-entropy.

2.1.2. Privacy Evolution. BC was designed with little privacy considerations initially. BC uses a fixed MAC address, which can be used as a unique identifier of a device. Even though only half of the MAC address (24 bits) is included in the data packet in plaintext, with a powerful sniffer, an attacker can recover the full MAC address of a BC device to enable user tracking [55]. A device's MAC address is critical to its privacy due to the uniqueness of its MAC address. As such, since Bluetooth 2.1, the non-discoverable mode of BC has been introduced to help reduce the chance of leaking the MAC address to an adversary. If a device is in the non-discoverable mode, it does not respond to inquiry requests, but it can still be connected from the devices that know its MAC address. This mode reduces the chance for an attacker to get a BC device's MAC address in the device discovery phase.

2.1.3. Status Quo of BC Security and Privacy. Since Bluetooth 4.1, the *Secure Connections Only* (SCO) mode has been introduced. In this mode, a device can only use state-of-the-art security features and FIPS-approved algorithms [56], i.e., SSP with authenticated pairing methods (e.g., NC and PE), secure authentication, and AES-CCM with a 128-bit-entropy encryption key. Thus, if both devices are Bluetooth 5.4-compliant and in the SCO mode, they are immune to all the above-mentioned attacks exploiting design flaws of the security features.

In terms of BC privacy, a device is vulnerable to tracking attacks if it is in use or is not in the non-discoverable mode. As such, the best practice in the real world is to set the device in the non-discoverable mode except during pairing. This is also the default setting on mainstream OSes, like iOS, Android, Linux, etc.

2.2. Bluetooth Low Energy

Bluetooth Low Energy (BLE), which is designed for power-constraint devices, also works in a central-peripheral scheme like BC. In the device discovery phase, the peripheral device keeps broadcasting advertising messages. The central device scans for the advertising messages to discover the peripheral device. Similar to BC, BLE also uses a pairing process for two devices to derive a shared secret key in the key sharing phase. In the data exchange phase, the data exchanged between the two devices is encrypted using the shared secret key to provide confidentiality guarantees. In the latest version of BLE (Bluetooth 5.4), a new feature named *advertising encryption* is introduced, allowing the central device to receive data with confidentiality guarantees from the peripheral device in the device discovery phase without establishing a connection.

2.2.1. Security Evolution. Unlike BC, BLE does not have an authentication procedure in the data exchange phase. As such, BLE's security involves two aspects: pairing and encryption (including data exchange encryption and advertising encryption).

Pairing. The first release of BLE (Bluetooth 4.0) uses a *legacy pairing* protocol for two devices to agree on a shared secret key. Like the legacy pairing of BC, BLE's legacy pairing protocol also does not use the DHKE protocol. Thus, it is also vulnerable to passive attacks, through which an attacker can recover the secret key if she can sniff the traffic of pairing [32].

To mitigate this weakness, since Bluetooth 4.2 [57], the *Secure Connections Pairing* (SCP) has been introduced. SCP is similar to the SSP of BC as they have the same steps, but different underlying cryptographic functions (e.g., SCP uses AES-CMAC while SSP uses HMAC-SHA-256). Because of the similarity, SCP also has the same weaknesses as SSP and the same mitigations for these weaknesses. Specifically, the weakness that allows an attacker to exploit fixed or reused passkeys [22]–[24] has been addressed since Bluetooth 5.1 [49, p.2450]. Also, since Bluetooth 5.1 [49, p.2446], attacks exploiting invalid public keys [15], [25] are no longer effective. Since version 5.3 [41, p.1578], the weakness enabling reflection attacks [21] has been fixed.

Data Exchange Encryption. Since the introduction of BLE, it uses AES-CCM as its encryption algorithm for data exchange. For this reason, the attacks against BC E0 encryption [27]–[31] are not effective for BLE. Additionally, the specification also requires that the minimum entropy of an encryption key is 7 bytes [50, p.1966]. However, the entropy is not large enough and can still be brute-forced [6].

Advertising Encryption. In Bluetooth 5.4, the SIG introduced a new security feature that allows advertising messages to be encrypted using AES-CCM [58, p.1351]. To use this feature, two devices need to pair first, and then the peripheral device shares a session key with the central device. The session key is protected by the data exchange encryption introduced above. After disconnection, the peripheral device broadcasts advertising messages encrypted using this session key. The central device can receive and decrypt the advertising messages because it has the same session key. With this feature, the fingerprinting attacks relying on plaintext advertising payload [33], [34] are no longer effective.

2.2.2. Privacy Evolution. Unlike BC, BLE has been designed with privacy considerations since its first release. While BC can use only one fixed MAC address, the privacy feature of BLE allows a device to use changeable MAC addresses. By switching to different MAC addresses frequently, a BLE device is difficult to track. The core of this privacy feature is the resolvable MAC address (RMA) mechanism [50, p.1733]. A BLE device can use the RMAes generated using a key [50, p.1733]. During pairing, the BLE device shares this key with a peer device. When the device uses RMAes, these MAC addresses seem purely random to other devices while the paired peer device can use the received key to resolve the RMAes [50, p.1744] and recognize the BLE device.

Since Bluetooth 5.0 [59], two privacy modes, *device privacy* and *network privacy*, have been introduced to further enhance BLE privacy. If a device is in the device privacy mode, it can only use changeable MAC addresses, and it can communicate with a device using a fixed MAC address. If a device is in the network privacy mode, both devices can only use changeable MAC addresses.

From Bluetooth 4.0 to 5.1, the specification requires that the *minimum* time interval between the MAC address changes is 15 minutes [49, p.2226], without an upper limit. As such, many BLE devices in the real world do not change their MAC addresses even for days [34], [60] compromising their privacy. To address this issue, since Bluetooth 5.2, the specification has required that the *maximum* interval between MAC address change is 15 minutes [54, p.1414].

2.2.3. Status Quo of BLE Security and Privacy. Since Bluetooth 4.2, BLE has also introduced the SCO mode, in which the SCP with authenticated pairing methods (e.g., NC and PE) and a 128-bit-entropy encryption key are used. Like BC, if both Bluetooth 5.4-compliant BLE devices are in the SCO mode, they are not affected by all the previously mentioned attacks.

For BLE privacy, except for the privacy issues caused by configuration or implementation [33], [34], [60]–[62], no weakness of the BLE privacy feature has been revealed.

2.3. Bluetooth Mesh

Though Bluetooth Mesh (Mesh) is built on BLE, it does not adopt BLE's central-peripheral scheme. Mesh utilizes BLE advertising to allow devices to form a mesh network that supports many-to-many communications.

In the device discovery phase, Mesh uses the same advertising mechanism as BLE to let devices discover each other. In the key sharing phase, instead of pairing, Mesh uses a configuration procedure to distribute secret keys to Mesh devices. In the data exchange phase, Mesh uses the shared secret keys to encrypt/decrypt the transmitted data.

2.3.1. Security Evolution. Since Mesh's introduction, its security features, which primarily involve two parts, configuration and encryption, have remained unchanged.

Configuration. A Mesh device must have two types of secret keys (i.e., *network key*, and *application key*) for encryption. A device obtains these keys via the following two steps. First, before joining a Mesh network, a Mesh device needs to be provisioned via the *provisioning* protocol [63, p.229]. During provisioning, the Mesh device receives a network key from the provisioner. Second, after joining the Mesh network, the Mesh device is further configured, usually by the provisioner, to run certain applications and to receive corresponding application keys (encrypted using the network key).

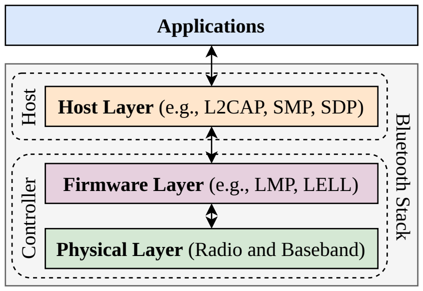
There are several known weaknesses in the provisioning protocol. The improper design of the authentication in the provisioning protocol allows for reflection attacks leading to device impersonation [13], [21]. The PIN used during provisioning may have a small entropy, e.g., 1 byte, and is vulnerable to brute-force attacks [21]. The calculation of the commitment value in the provisioning protocol is malleable resulting in PIN leakage [21]. An attacker can launch any of the three attacks to get the network key and application key during the configuration process.
Encryption. In the data exchange phase, Mesh requires two layers of encryption, the network layer using an encryption key derived from the network key and the application layer using the application key. All devices within the same network or running the same application share the same network key and application key. As such, only the Mesh device within the same network running the same application can decrypt the messages. Mesh also uses AES-CCM as its encryption algorithm and mandates a 128-bit-entropy encryption key [63, p.101].

2.3.2. Privacy Evolution. Unlike BC/BLE, Mesh does not transmit source or destination MAC addresses in plaintext. It encrypts the destination MAC address together with the payload and obfuscates the source MAC address using a privacy key derived from the network key. By encrypting and obfuscating the MAC addresses, it is difficult to track a Mesh device.

2.3.3. Status Quo of Mesh Security and Privacy. In the latest version of Mesh specification [63] at the time of the paper writing, the weaknesses of the provisioning protocol mentioned earlier still exist. However, these weaknesses are addressed via advisories from the SIG [36], [37], [64].

2.4. Observations

Bluetooth is getting more secure. Though BC, BLE, and Mesh all have their corresponding security features when they were first released, from BC to Mesh, Bluetooth is getting more secure across protocols. For example, when BC was introduced, it uses the E0 stream cipher, which is vulnerable to the correlation attacks [27]–[31]. BLE uses the AES-CCM encryption that is immune to these attacks since its first release. In Mesh, the entropy of the encryption key has to be 128 bits, preventing brute-force attacks [6] that may affect BLE. Besides, Bluetooth is also getting more secure within one protocol. For instance, the SSP and the SCP were introduced to replace BC's and BLE's legacy pairing protocols respectively to fundamentally address the design weaknesses in the legacy protocols. Later, both BC and BLE introduced the SCO mode to further enhance their security by allowing only their latest security features.
Bluetooth is getting more private. From the firstly introduced BC to the recently introduced Mesh, Bluetooth focuses more on device privacy. BC is not designed with privacy considerations since it uses fixed MAC addresses in plaintext, allowing naive device tracking based on MAC addresses. BLE is designed with privacy considerations that a BLE device can change its MAC addresses and thus can prevent naive tracking. However, its MAC address is still in plaintext and can be tracked if the device does not change its MAC address frequently. In Mesh, the MAC addresses are either encrypted or obfuscated and are not in plaintext anymore, making naive tracking ineffective.



Applications
Host
Host Layer (e.g., L2CAP, SMP, SDP)
Controller
Firmware Layer (e.g., LMP, LELL)
Physical Layer (Radio and Baseband)
Bluetooth Stack

Figure 1: Layers of a Bluetooth stack.

Backward compatibility comes before security. BC and BLE have introduced new security features (i.e., SSP, SCP, and secure authentication) to replace the legacy ones (i.e., legacy pairing and legacy authentication). Though the new security features are more secure than the legacy ones, the legacy features are still supported due to compatibility considerations. Similarly, to be compatible with devices that do not support a large-entropy encryption key, SSP and SCP still allow to use an encryption key with less than 16-byte entropy (albeit larger than 7-byte). Likewise, even though a device can be configured to use the SCO mode to be more secure, to the best of our knowledge, we are not aware of any general-purpose devices (e.g., smartphones or laptops) using this mode by default. The reason is that in the SCO mode, a device cannot pair and communicate with another device that only supports JW, such as a headset or a speaker.

3. Systematization Matrix

As shown in Figure 1, Bluetooth is composed of a *controller stack* and a *host stack*. The code of these two components usually executes on different microcontrollers (MCU)[1]. The controller stack code runs on an MCU within a Bluetooth chip, while the host stack code usually runs on the main MCU together with the host OS (e.g., Linux). We divide the Bluetooth stack into three *layers* based on their functionalities and the location where the functionalities should be implemented based on the specification. The *physical layer* is located at the bottom of the Bluetooth stack and is responsible for the physical signal process. The *firmware layer* lies in the controller on top of the physical layer, establishing link-layer connections and transmitting and processing data between the physical layer and the host layer. The *host layer* represents all components in the host stack, including the protocols for data exchange on the host stack, such as the Logical Link Control and Adaptation Protocol (L2CAP) [41, p. 1011], and components to finish specific tasks on the host stack, such as Bluetooth profiles [65].

To study existing attacks and defenses of Bluetooth and summarize their similarities and differences, we first conduct a survey of Bluetooth attacks and defenses of existing research covering all the papers published in the big four top-tier conferences (i.e., S&P, USENIX Security, CCS, and NDSS) from 2000 to March 2023. We also select papers from academic

1. On embedded devices, the host and controller stacks may be executed on the same MCU.

conferences and journals that are broadly related to Bluetooth security and privacy (including but not limited to MobiSys, WiSec, PETS, and WOOT). For industry conferences, we focused on the three most reputable ones: Black Hat, DEF CON, and CCC. We select papers from these venues using the following criteria based on our judgment. For attacks, we select papers revealing *previously-unknown* Bluetooth security and privacy issues. We skip papers discussing application-level vulnerabilities that are not caused by Bluetooth itself (e.g., problems in device tracking protocols based on BLE [66]). For defenses, we include papers talking about detecting or mitigating existing Bluetooth attacks. Lastly, we also include attacks that are not covered in the above but are reported by The Hacker News [67] (the most popular cybersecurity media according to FeedSpot [68]).

To further understand the capabilities of the attacks and the scope of the proposed defenses, we classify the surveyed attacks and defenses by their affected layers. At each layer, we further group attacks and defenses into subcategories based on the techniques used, as we will discuss in Section 4 and Section 6. Moreover, for each attack/defense, we analyze its affected protocols (i.e., BC, BLE, and Mesh), affected phases (i.e., device discovery, key sharing, and data exchange), and attack model (i.e., Dolev-Yao [69], whether the device is compromised, and whether user mistakes are required).

To better reflect the cost of attacks and defenses, we divide each attack model into more fine-grained categories. In the Dolev-Yao attack model, we further investigate the hardware required by each implementation of attacks and defenses. Moving from stronger towards weaker capabilities, we study if an attack or defense requires Software Defined Radios (SDR, e.g., USRP B210 [70], around $2,000), Programmable-Firmware Devices (PFD, the device of which the firmware is programmable, e.g., Ubertooth [71], around $150), or Generic Bluetooth Adapters (GBA, e.g., USB Bluetooth adapter [72], around $15). In the attack model where a communicating device is compromised, we study the permissions required by each attack and defense.

In general, attacks and defenses targeting BLE and Mesh require weaker hardware capabilities compared with BC. All BLE devices, including GBAs, can inject and eavesdrop on arbitrary advertising messages in the device discovery phase of BLE. Therefore, GBAs are enough for attacks or defenses targeting the data in BLE advertising messages (i.e., MAC addresses or payloads) in the device discovery phase. However, GBAs are not powerful enough to obtain the physical features of BLE advertising messages, nor can they eavesdrop on or inject messages in the other two phases (i.e., key sharing and data exchange) because of the Frequency Hopping (FH) mechanism in these two phases. Therefore, attacks or defenses relying on physical features (e.g., accurate timing [73] or carrier frequency offset [74]) of BLE advertising messages or targeting the other two phases (e.g., data exchange [75]) require SDRs or PFDs. Attacks or defenses at the physical layer of Mesh have the same requirements as BLE since Mesh is built upon the physical layer of BLE. For BC, GBAs cannot inject or eavesdrop on arbitrary messages (i.e., inquiry scan and response) during the device discovery phase. Accordingly, attacks or defenses of BC discovery phase require SDRs or PFDs (e.g., Blue's clue [76]). BC also uses FH in the key sharing and data exchange phases resulting in the need for SDRs or PFDs for attacks and defenses in these two phases [55]. We will refer to these three types of hardware capabilities when we study each attack in Table 2 and defense in Table 3.

Lastly, we investigate the relationship between attacks and proposed defenses (Section 7) based on the analysis and layered categorization, which allows us to find the missing parts of current defenses (Section 8). It can also guide the future direction to make Bluetooth more secure (Section 9).

4. Systematization of Attacks

In this section, we present the attacks against Bluetooth we studied during the survey.

4.1. Physical Layer

Since Bluetooth's physical signal is transmitted Over The Air (OTA), which is an accessible medium to an attacker within physical proximity, attacks against the physical layer can be grouped into two categories, *signal eavesdropping* and *signal injection*.

4.1.1. Signal Eavesdropping. BC employs a FH spread spectrum mechanism, in which the hopping sequence is determined by the central device's MAC address and clock, to avoid signal interference. The central device's MAC address and clock are available only during the device discovery period. As such, in the data exchange phase, the central device's (if non-discoverable) MAC address and clock are unknown to an attacker. Accordingly, FH can prevent an attacker from eavesdropping on BC communications since the attacker does not know the hopping sequence and cannot hop along with the communicating devices to receive the signal.

However, despite the unavailability of the central device's MAC address and clock to an attacker, she can either brute-force [77] or infer [55], [78] the central device's MAC address and clock to figure out the hopping sequence. As a result, the attacker can further eavesdrop on unencrypted data [78] or track the user [55], compromising the user's privacy. An attacker can also eavesdrop on the whole Bluetooth bandwidth to conduct relay attacks [79].

Since BLE also adopts a similar FH mechanism to combat interference in the data exchange phase, an attacker can also brute-force the parameter to calculate the hopping sequence to enable eavesdropping [78]. Once the data is eavesdropped, though encrypted, the device user's privacy can be leaked. For example, with the captured BLE traffic from a fitness tracker, the attacker can further detect the current activity of the user [60].

In addition, in the device discovery phase of BLE, the peripheral device proactively broadcasts advertising messages containing its MAC address. As such, an attacker can sniff the peripheral device's advertising messages to track it and its user if it uses an unchanged MAC address [33], [80]. In the case where the peripheral device frequently changes its MAC address, the attacker can still identify and track a BLE device either via fingerprinting [74] or by linking its BLE advertising messages to its unique BC MAC address if it is BC/BLE dual-stack [73]. Besides BLE itself, there are protocols that are built upon BLE advertising, such as Apple's Find My [81] and Continuity [82] protocols. If these protocols are not carefully designed to preserve privacy, devices running these protocols can be easily tracked because of BLE's proactively broadcasting nature. For example, even with BLE's privacy features enabled, an attacker can still track these devices by exploiting the location reports in the Find My protocol [83], the usage of usernames in BLE's advertising messages [84], or the usage of predictable sequence numbers [85], [86].

Since Mesh is built upon BLE, theoretically, the attacks against BLE can potentially affect Mesh. However, due to Mesh's different addressing mechanism (MAC addresses are either encrypted or obfuscated) [63, p.23], attacks relying on BLE's MAC address [33], [80] or targeting protocols that are based on BLE [83]–[86] are not effective against Mesh while other BLE eavesdropping attacks [60], [73], [74], [78] are potentially effective against Mesh.

4.1.2. Signal Injection. As mentioned in Section 2.1.2, a BC device does not respond to inquiry requests while it still responds to connection requests if it is in the non-discoverable mode. Since initiating a connection request only requires the MAC address, an attacker can inject probing connection requests brute-forcing all the MAC address space. Therefore, the attacker can discover a nearby device that is in the non-discoverable mode if she can receive responses [76], [87].

Similar to BC, the probing method also affects BLE privacy. An attacker can inject probing BLE advertising messages with specific MAC addresses. If the victim device uses the "filter accept list" [58, p.1314] feature that allows the victim device to automatically respond to the advertising messages from devices on the list. The attacker can infer that the victim device is close to the attacker, if she receives responses for the probing advertising messages, violating the victim device's privacy [62].

Besides privacy, signal injection also affects security. The specification (from Bluetooth 4.2 to 5.2) allows a BLE device to expand the receiving time window to compensate for clock inaccuracy. As such, an attacker can exploit this feature to increase the chance to inject malicious traffic into an established BLE connection to perform MitM attacks [75]. Additionally, since Bluetooth 5.0, the FH sequence of BLE can be guessed, leading to jamming attacks against BLE [88].

4.2. Firmware Layer

Firmware is a piece of software running on an MCU of a Bluetooth chip. Like other software, it may have implementation errors that can be exploited by attackers. We categorize attacks against the firmware at the *implementation level* into the *firmware exploitation* subcategory. Since the firmware implements BC/BLE/Mesh's security features discussed in Section 2[2], it is also affected by attacks exploiting *design-level* weaknesses of these features. We categorize such attacks into three subcategories, namely *breaking key sharing*, *authentication bypass*, and *breaking encryption*, based on the targeted feature of the attack.

2. BC pairing, authentication, encryption, and BLE encryption are implemented at the firmware layer. The BLE pairing, Mesh configuration, and Mesh encryption reside at the host layer. However, due to the similarity of these features, we include the attacks against BLE pairing and Mesh configuration and encryption into the firmware layer to simplify the description.

4.2.1. Firmware Exploitation. Since the firmware lies between the physical layer and the host layer, it handles inputs from both directions (i.e., OTA packets from the physical layer and data from the host layer). Accordingly, the firmware can be exploited both OTA and locally.

An attacker can launch attacks like Denial of Service (DoS) [89]–[91] and Remote Code Execution (RCE) [4], [10], [11], [92] OTA against a victim device. Attacking the firmware OTA is stealthier, and it needs PFDs or SDRs. This is because these attacks need to send malformed messages to trigger vulnerabilities within the firmware, which GBAs usually cannot do. Both BC and BLE have their corresponding firmware implementations, and thus can be affected by firmware exploitation attacks. Since Mesh relies on BLE's firmware implementation, the attacks affecting BLE [4], [11], [89] can also potentially affect Mesh. Besides attacking the firmware OTA, if an attacker can interact with the firmware from the host layer (i.e., has the permission to send data to the firmware), she can also exploit the firmware locally [91], [93].

4.2.2. Breaking Key Sharing. Attacks in this subcategory exploit the design weakness of the protocols in the key sharing phase, such as the SSP of BC, the SCP of BLE, and the Mesh provisioning protocol. To perform these attacks, an attacker needs to conduct selective jamming or sniffing, and thus, PFDs or SDRs are required. Most of the attacks in this category have been introduced in Section 2, including the attacks against BC pairing [15], [18]–[25], [43]–[48], BLE pairing [15], [21], [25], [32], and Mesh provisioning [13], [21]. All the above attacks assume the classic Dolev-Yao attack model without requiring the user's mistakes in the key sharing phase.

BThack [94] and PE pairing method confusion attacks [95], [96] prove that with a minor user mistake during BC/BLE pairing, which is caused by the confusion of used pairing methods, an attacker can launch MitM attacks against the two devices. What is worse, users are likely to make mistakes since they are not well-notified during pairing [97]. BThack exploits the confusion of the NC and PE pairing methods, while PE pairing mode confusion attacks exploit the confusion of the legacy PE pairing method and the PE pairing method in SSP or SCP. BThack is effective even if both devices are in the SCO mode, and PE pairing mode confusion attacks are still effective when one device is in the SCO mode. Compared to other attacks against the key sharing phase, the SIG did not revise the specification to fix the issues causing these attacks. Instead, it issued advisories suggesting preventing these attacks by improving the user interface (UI) design to reduce the user's chance of making mistakes [95], [96], [98]. The reason behind this, we believe, is that the specification assumes the Dolev-Yao attack model only [41, p.268], in which the user does not make mistakes.

4.2.3. Authentication Bypass. Since only BC implements the authentication procedure in the data exchange phase, attacks in this category [7], [99] only affect BC. As mentioned in Section 2.1.1, bypassing the BC authentication itself has a limited impact. This is because two BC devices will enable encryption after the authentication. Without breaking the encryption, an attacker cannot inject/obtain any meaningful traffic into/from

TABLE 2: Attacks grouped into layers and attack types, and their corresponding goal, affected protocol, phase, and attack model. DD: device discovery, KS: key sharing, DE: data exchange, DY: Dolev-Yao, DC: device is compromised, UM: user mistake.

Layers	Attack Type	Attack	Malicious Goal	Affected Protocol			Affected Phase			Attack Model		
				BC	BLE	Mesh	DD	KS	DE	DY	DC	UM
Physical Layer	Signal Eavesdropping	BlueSniff [77], Full-band sniffer [55]	Prv	●				◐	●	*S*		
		User identification [80], Device tracking [33]	Prv		●		●			*G*		
		Link BC/BLE [73], Location tracking [74]	Prv		●	◐	●			*S*		
		User activity detection [60]	Prv		●	◐	●		●	*S*		
		Device tracking [83]–[86]	Prv		●		●			*G*		
		BlueEar [78]	Prv	●	●	◐	●	◐	●	*P*		
		Relay attack [79]	Sec	◐	●	◐	●	◐	●	*S*		
	Signal Injection	Discover non-discoverable device [76], [87]	Prv	●			●			*S*		
		Allowlist-based device tracking [62]	Prv		●		●			*G*		
		InjectaBLE [75], Btlejack [88]	Sec		●	◐		◐	●	*P*		
Firmware Layer	Firmware Exploitation	InternalBlue [10]	Sec	●				●		*P*		
		BrakTooth [90]	Sec	●			●	●	●	*P*		
		BleedingBit [4], JackBNimBLE [92]	Sec		●	◐	●			*P*		
		SweynTooth [89]	Sec		●	◐		●	●	*P*		
		Frankenstein [11]	Sec	●	●	◐	●	◐	●	*P*		
		Attack wireless coexistence [91], [93]	Sec	●	●	◐	●	◐	●		*R*	
	Breaking Key Sharing	PIN brute-force [18]–[20], [100]	Sec	●				●		*P*		
		Key recovery [32]	Sec		●			●		*P*		
		Just Works abuse [43]–[48], Exploit passkey reuse [22]–[24]	Sec	●	◐			●		*P*		
		Invalid curve attack [15], [25]	Sec	●	●			●		*P*		
		BThack [94], PE pairing confusion [95], [96], MitM [97]	Sec	●	●			●		*P*		◇
		BlueMAN [13]	Sec			●		●		*P*		
		BlueMirror [21]	Sec	●	●	●		●		*P*		
	Authentication Bypass	Relay attack [99], BIAS [7]	Sec	●					●	*P*		
	Breaking Encryption	Correlation attack [27]–[31]	Sec	●					●		*H*	
		KNOB [5], [6]	Sec	●	●				●	*P*		
Host Layer	Host Exploitation	BlueBorne [3], BlueFrag [101]	Sec	●					●	*G*		
		SweynTooth [89]	Sec		●	◐		●	●	*P*		
		BrokenMesh [102]	Sec			●		●	●	*P*		
		ToothPicker [103]	Sec	●	●	◐		●	●	*G*		
		BleedingTooth [104]	Sec	●	●	◐	●		●	*G*		
	Illegal Service Access	Device mis-binding [105]	Sec	●					●		*B*	
		BLAP [106]	Sec	●					●	*G*	*R*	
		BadBluetooth [107]	Sec	●					●	*G*	*B*	
		Blacktooth [108]	Sec	●					●	*P*		
		Co-located app attack [109]	Sec		●				●		*B*	
		BLESA [8], Bluedoor [110], Downgrade attack [111], [112]	Sec		●				●	*G*		
		Toxic Friends [113]	Sec			●			●		*R*	
		CSIA [13]	Sec	●	●				●		*B*	
		BLURTooth [114]	Sec	●	●				●	*P*		◇
	Privacy Violation	Device tracking [33], Sensitive data leakage [34]	Prv		●		●			*G*		

●: Target protocol/phase of an attack. ◐: The protocol/phase that is not targeted by an attack but can also be affected by the same attack.
S: SDRs required (e.g., USRP B210 [70]). *P*: PFDs required (e.g., Ubertooth [71]). *G*: GBAs required (e.g., USB Bluetooth adapter [72]).
R: Root permission required. *H*: Hardware modifications required. *B*: Bluetooth permissions required.

a BC connection. As such, the authentication procedure is redundant given the protection provided by the encryption. This might be the reason that the latter introduced BLE and Mesh do not have similar authentication procedures to BC's. Like attacks in Section 4.2.2, attacks in this category also require selective jamming or sniffing capabilities, and thus need PFDs or SDRs.

4.2.4. Breaking Encryption.

The attacks against Bluetooth encryption [5], [6], [27]–[31] have been introduced in Section 2.1.1 and Section 2.2.1. Among these attacks, the KNOB attacks [5], [6] assume the Dolev-Yao attack model and require PFDs or SDRs to perform selective jamming and sniffing.

The correlation attacks [27]–[31], on the other hand, have different assumptions that an attacker can obtain the key stream of the E0 stream cipher. We believe it is not a realistic assumption because the key stream generation is implemented in the hardware. Since the key stream is not stored or output to the firmware [115], [116], obtaining the key stream will rely on either hardware modifications or side-channel attacks. As such, we believe this is the reason why no real-world attacks against the E0 cipher have been discovered, and why the E0 cipher is still supported in the 5.4 version of the specification, though the E0 cipher is not secure.

4.3. Host Layer

The host layer of Bluetooth is usually integrated into the OS, such as the BlueZ [117] stack on Linux. Inevitably, the host part of Bluetooth may contain implementation bugs that can be exploited by an attacker. In addition, Bluetooth services, such as the ones for keyboard and headset connection, are implemented at the host layer. Without proper access control, these services can be illegally accessed by an attacker. Moreover, the configuration and application-level data that may leak a user's private information (e.g., health information) is also set at the host layer. As such, we group the attacks targeting the host layer into three subcategories, *host exploitation*, *illegal service access*, and *privacy violation*.

4.3.1. Host Exploitation.

The host layer handles inputs received from an OTA device through the firmware layer. Accordingly, if the host layer contains vulnerabilities, an attacker can send malicious messages OTA to trigger these vulnerabilities, causing DoS [89], [103] or RCE [3], [101], [104] attacks that affect both BC and BLE. Since Mesh allows using BLE's Generic Attribute Protocol (GATT) as a bearer [63, p.28], the attacks affecting BLE [89], [103], [104] can also potentially affect Mesh. Besides sharing protocols with BLE, Mesh also has its unique ones (e.g., the provisioning protocol and network management protocols).

Attacks against these protocols [102] are unique to Mesh. In general, GBAs can help craft the malicious payload to exploit the host layer (e.g., BlueBorne [3]), while using PFDs can provide more flexibility for the attacks (e.g., SweynTooth [89]).

Compared to the firmware exploitation attacks, the host exploitation attacks may cause a much larger impact. The reason is that the firmware exploitation attacks may only affect the Bluetooth functionality since the firmware runs on the Bluetooth chip, while the host exploitation attacks can affect the whole OS because the host layer of Bluetooth can run in the OS's privileged mode. For example, part of BlueZ runs within the Linux kernel [118]. As such, the attacks targeting BlueZ kernel code [3], [104] can affect the whole Linux kernel rather than Bluetooth functionality only.

4.3.2. Illegal Service Access. The attacks in this subcategory target the data exchange phase of Bluetooth communication and allow an attacker to illegally access the services on the victim device either from a malicious device impersonating the legitimate paired peer device [8], [106], [107], [110], [111], [114] or from a malicious component (e.g., a malicious app on an Android phone) of the legitimate paired peer device [13], [105], [109]. Accordingly, these two types of illegal access attacks have different assumptions. Accessing services from a malicious device follows the Dolev-Yao attack model (e.g., BLESA [8] and Blacktooth [108]), while accessing services from a malicious component assumes that one of the paired devices is compromised (e.g., installing a malicious app [13], [105], [109]).

To access the victim device's services from a malicious device, the malicious device can first impersonate the legitimate paired peer device and re-pair with the victim device via either the assistance from a malicious app [106], [107] or the user's mistakes [114]. As such, the new pairing with the malicious device overwrites the pairing with the legitimate peer device, allowing the malicious device to access the victim device's services. An attacker can also impersonate the legitimate paired peer device by exploiting design or implementation vulnerabilities [8], [108], [110], [111] of the data exchange phase without pairing with the attacker to access the victim device's services from a malicious device. To access the victim device's services from a malicious component of a legitimate paired peer device, an attacker can use the shared secret key on the legitimate peer device to bypass the protection provided by authentication and encryption [13], [105], [109].

By accessing the victim device's services, the attacker can inject keyboard strokes [107] or spoofed data [8] into the victim device, read sensitive data from the victim device [105], [109], or even perform MitM attacks [111]. Since Mesh introduces an extra layer of encryption at the application level, these illegal service access attacks against BC and BLE do not affect Mesh.

4.3.3. Privacy Violation. Despite the privacy feature presented in Section 2.2.2, improper configuration at the host level can still undermine a device's privacy. If the device is not properly configured by the upper-layer application, the device may include unique identifiers or personal information in the advertising messages of BLE. An attacker can either track the BLE device and its user leveraging these unique identifiers [33] or obtain the user's personal information [34]. GBAs can receive BLE advertising messages, and thus are capable enough to launch attacks in this category.

5. Attacks Takeaways

By analyzing existing attacks, as systematized in Table 2, we found common trends regarding Bluetooth privacy and security issues. Based on these trends, we have the following findings:

T1. BLE device discovery phase draws most privacy attacks. Because of the unique device discovery mechanism of BLE (i.e., BLE advertising), attacks against BLE privacy focus on the device discovery phase. On the one hand, an attacker can trivially receive advertising messages from a peripheral device. The peripheral BLE device proactively broadcasts advertising messages in the device discovery phase even though no device around intends to discover it. On the other hand, these advertising messages may contain private information, such as unique MAC addresses and usernames, leading to tracking attacks [33], [80] or privacy leakage [34]. Besides the device discovery phase, the other two phases may raise privacy issues and should also be scrutinized, as we will further discuss in Section 9.

T2. Pairing security faces challenges caused by users' mistakes. Since a user's confirmation is required during pairing, if she is confused by which pairing method is used, she may mistakenly confirm the pairing, resulting in MitM attacks [94]. Even worse, the SCO mode cannot prevent such attacks [94]–[96]. Fundamentally addressing these issues requires protocol modifications that will introduce compatibility concerns (e.g., the proposed defense against BThack [94]). The current defense against these attacks relies on the user making no mistakes during pairing. However, even for developers, it is not straightforward to understand which method should be used during pairing, not to mention the general users [94]. As a result, it is challenging to rely on the user's right decision to secure Bluetooth pairing.

T3. Bluetooth specification has different assumptions from its implementations on modern OSes. When two paired devices reconnect, the specification assumes the Dolev-Yao model, in which the two devices are not compromised. However, in the real world, a device could have been (partially) compromised (e.g., a malicious app running on an Android phone). In fact, modern OSes are designed under the assumption that an app could be malicious [119]. For this reason, the OSes isolate each app using sandbox mechanisms. The inconsistency of assumptions leads to the attack that a malicious app on an Android phone with Bluetooth permissions can invoke Android APIs to read/write data from/to any Bluetooth devices that are paired with the phone [105], [109].

Because of this assumption inconsistency, the attacks [13], [105], [107], [109] are not practical attack scenarios according to the specification, but they pose a legitimate threat in the real world. To the best of our knowledge, none of these attacks can be prevented by specification updates. As such, it relies on the developers of the general-purpose or IoT devices to implement mitigations against these attacks (e.g., an additional access control mechanism or application-level authentication or encryption). One caveat for developers is to consider if their system's assumptions are in line with the specification's.

If not, developers should consider implementing necessary security mechanisms to prevent potential issues caused by the inconsistency of assumptions.

6. Systematization of Defenses

Over the years, researchers have proposed multiple defensive mechanisms to improve the security of Bluetooth. Similar to the systematization for attacks, we study the target protocol, phase, and attack model for each defense. To better understand the feasibility of a defense, we also investigate the hardware (i.e., whether extra SDRs, PFDs, or GBAs are required) and software (i.e., the permission required to deploy the defense on a target device) prerequisites of a defense. Lastly, we categorize defenses into layers based on the layer they are designed to protect and group them into subcategories based on the defensive technique adopted.

6.1. Physical Layer

Defenses at the physical layer leverage the physical features of a device's Bluetooth signal to detect or prevent signal injection and signal eavesdropping. In general, there are two stages in a defense, a fingerprinting stage and an identification stage. The defense first fingerprints authorized devices by learning their unique physical features, such as the clock skew [12], [120] and signal strength [12], [121]. Then, in the identification stage, the defense keeps collecting the physical features of the communicating devices and checking whether these physical features are consistent with the fingerprints of the authorized devices. If they are inconsistent, it means signals from unauthorized devices are detected. The defense can warn the user to defend against signal injection attacks, such as BlueID [120] and BlueShield [12]. If they are consistent, it means the signal is from an authorized device. In this case, the defense can jam the communication channel so that an attacker cannot receive the signal from the authorized device to defend against signal eavesdropping attacks, such as BLE-guardian [34].

6.2. Firmware Layer

Like securing other protocols, at the implementation level, researchers have proposed approaches to fuzz and analyze the firmware (*firmware fuzzing and analysis*), through which implementation vulnerabilities can be revealed and further patched. At the design level, *formal verification*, an effective automatic method to detect design weaknesses, has also been adopted by researchers to either discover design weaknesses in the specification or provide security guarantees for the security protocols. Finally, to mitigate the issues caused by protocol design weaknesses, researchers have proposed enhancements (*protocol enhancement*) to fix the design weakness in the protocol.

6.2.1. Firmware Fuzzing and Analysis. Prior literature [122]–[125] has proven the effectiveness of firmware fuzzing and analysis to find firmware vulnerabilities. The same techniques are also applicable to Bluetooth firmware.

Fuzzing techniques can be adopted to discover memory corruption bugs in Bluetooth firmware. The fuzz input could be generated from either a real device [10], [89], [90], [126] or an emulated device [11]. In general, on the one hand, using a real device to provide fuzz input can be easily ported to fuzz the firmware of different controllers, while implementing the emulated device for different controllers' firmware needs extra effort. On the other hand, providing fuzz input from an emulated device has better performance since the input is not transmitted OTA. Due to the closed-source nature of the firmware, firmware fuzzing requires either reverse-engineering the device's firmware (e.g., InternalBlue [10] and Frankenstein [11]) or self-implementing a piece of firmware (e.g., SweynTooth [89]).

Different from fuzzing, which needs to run the firmware, statically analyzing the firmware [127], [128] can discover security or privacy issues that are caused by insecure configurations without running the firmware. For example, through static analysis, the insecure configuration that a BLE device can transmit data in plaintext allowing a passive attacker to eavesdrop on the data can be detected [128]. In addition to directly discovering security issues, firmware analysis can help enhance firmware security, for example, by debloating [129].

6.2.2. Formal Verification. While firmware fuzzing and analysis aims at automatically detecting issues in specific firmware *implementations*, formal verification attempts to detect *design* issues in the Bluetooth specification. Meanwhile, due to the development of automatic formal verification tools, such as ProVerif [130] and Tamarin [131], *automatic* formal verification of Bluetooth protocols is feasible and efficient.

Since key sharing protocols (e.g., SSP) are the cornerstones of Bluetooth security, they have drawn the most attention from researchers. Due to the complexity of key sharing protocols, researchers have different focuses on their modeling and analyses, for example, a sub-protocol of a pairing method (e.g., the Diffie-Hellman protocol [15]) or one specific pairing method (e.g., NC in SSP [132] and PE in SCP [14]). Inspired by the attack exploiting two different pairing methods [94], researchers have also verified key sharing protocols in a more comprehensive manner considering combinations of protocols, for example, the analysis of BC SSP [13], BLE SCP [133], and Mesh provisioning [13]. Besides the key sharing protocols, researchers have also verified the authentication and encryption procedures in the data exchange phase and identified issues that allow spoofing [8] and illegal access attacks [13].

While automatic formal verification is effective in detecting design weaknesses, the trade-off between the modeling granularity and coverage needs to be carefully considered by researchers to make the model tractable. Finer-grained modeling may be able to discover more attacks, but it has to sacrifice the model coverage (e.g., modeling and verifying part of a whole protocol [14], [15]) to be tractable. On the contrary, for better model coverage, researchers may need to adopt a coarser-grained modeling choice for one part of the whole protocol to make the model tractable [13].

6.2.3. Protocol Enhancement. To address certain design weaknesses in Bluetooth protocols, different enhancements to these protocols have been proposed. Existing enhancements focus on two areas: BLE advertising and BC/BLE pairing.

TABLE 3: Proposed defenses grouped into layers, used techniques, security goals, protected protocols, phases, and attack models of the defense. DD: device discovery, KS: key sharing, DE: data exchange, DY: Dolev-Yao, DC: device is compromised.

Layer	Defense Technique	Defense	Goal	Prerequisites		Protected Protocol			Protected Phase			Attack Model	
				HW†	SW‡	BC	BLE	Mesh	DD	KS	DE	DY+	DC★
Physical Layer	Device Identification	BlueID [120]	Sec	*P*		●			◐	◐	●	*G*	
		BLE-guardian [34]	Prv	*P*	*B*		●	◐	●			*G*	
		BlueShield [12]	Sec	*P*			●	◐	●			*G*	
		Relay attack detection [121]	Sec		*B*		●	◐	●			*G*	
Firmware Layer	Firmware Fuzzing and Analysis	BrakTooth [90]	Sec	*P*		●			●	●	●	*P*	
		Sweyntooth [89], BLEDiff [134]	Sec	*P*			●	◐	●	●	●	*P*	
		FirmXRay [128]	Sec&Prv				●	◐	●	●	●	*G*	
		ProXray [127]	Sec			●	●	◐	●	●	●	*G*	
		InternalBlue [10], InsideJob [126]	Sec	*P*		●	●	◐	●	●	●	*P*	
		Frankenstein [11], LightBlue [129]	Sec		*R*	●	●	◐	●	●	●	*P*	
	Formal Verification	Formal analysis of SSP [132], [135], [136]	Sec			●				●		*S*	
		Formal analysis of BLE reconnection [8]	Sec				●				●	*S*	
		Formal analysis of SCP [133] and PE in SCP [14]	Sec				●			●		*S*	
		Formal analysis of Diffie-Hellman [15]	Sec			●	●	◐		●		*S*	
		Comprehensive Bluetooth formal analysis [13]	Sec			●	●	●		●	●	*S*	*B*
	Protocol Enhancement	Advertising enhancement [137], Eddystone [138]	Sec		*R*		●		●			*G*	
		MagicPairing [139]	Sec		*R*	●	●			●	●	*S*	
Host Layer	Host Fuzzing and Analysis	BLEScope [140]	Sec&Prv				●		●	●	●	*G*	
		SweynTooth [89], BLEDiff [134]	Sec	*P*			●		●	●	●	*P*	
		L2Fuzz [141]	Sec	*P*		●					●	*P*	
		Frankenstein [11], ToothPicker [103]	Sec		*R*	●	●		●	●	●	*P*	
		ProXray [127]	Sec			●	●		●	●	●	*G*	
		BLE Mesh fuzzer [102]	Sec	*P*				●		●	●	*G*	
	Traffic Analysis and Filtering	Intrusion detection [142]	Sec	*S*		●					●	*G*	
		LBM [9]	Sec		*R*	●	●		●		●	*G*	
		Interaction verification [143]	Sec		*R*	●	●		●	●	●	*G*	*B*
	System Hardening	Dabinder [105], Profile binding [107]	Sec		*R*	●					●		*B*
		Application-level security [109]	Sec		*R*		●				●		*B*
		LightBlue [129]	Sec		*R*	●	●				●	*G*	

●: Target protocol/phase of a defense. ◐: The protocol/phase that is not targeted by a defense but can also be protected by the same defense.
†: *P* and *S* represent that a defense requires extra PFDs or SDRs, respectively. ‡: *B* and *R* indicate that a defense requires Bluetooth or the Root permission to deploy, respectively.
+: *S*, *P*, and *G* indicate that a defense assumes attackers to use SDRs, PFDs, or GBAs to attack, respectively. ★: *B* represents that a defense assumes attackers to have Bluetooth permissions.

To improve the security of BLE advertising, Wang et al. introduced a counter in advertising messages and proposed a three-way handshake scheme [137] to enhance BLE advertising and defend against replay attacks. Google developed Eddystone [138] which encrypts advertising messages so that only authorized devices can decrypt them. Unfortunately, this system is obsoleted at the time of the paper writing. Nevertheless, in Bluetooth 5.4, a BLE device can encrypt its advertising messages, as discussed in Section 2.2.1. Apple implemented MagicPairing [139] to enhance the BC/BLE pairing protocol between Apple devices. It relies on the iCloud service to synchronize keys, based on which the Bluetooth secret key is derived, among paired devices. For each communication, a fresh secret key is generated. By doing this, MagicPairing can achieve forward secrecy [144] that the specification does not provide.

Though enhancing the protocol can address its design weaknesses fundamentally, these enhancements face compatibility issues since they require modifications of devices on both ends of the communication. For this reason, to the best of our knowledge, the three-way handshake scheme is not adopted by real-world devices, and MagicPairing is enabled only among Apple products.

6.3. Host Layer

The firmware layer and the host layer share some defense techniques. The host code can also be secured via fuzzing and static analysis (*host fuzzing and analysis*). Moreover, Bluetooth communication at the host layer can also be enhanced (*system hardening*). Different from the enhancements at the firmware layer, which require protocol modifications introducing compatibility issues, enhancements at the host layer do not need protocol changes and usually can be deployed at one end of the communication.

There are also different defense techniques between the firmware layer and the host layer. Potentially, formal verification is also applicable at the host layer. However, none of the existing formal verification can be categorized into the host layer because the target protocols of existing formal analyses are implemented in the firmware. Additionally, *traffic analysis and filtering*, the technique employed in securing other network protocols [145], [146], has been applied to Bluetooth at the host layer while not applied at the firmware layer. The reason might be the closed-source nature of the firmware, which makes implementing traffic analysis mechanisms in the firmware challenging. Additionally, the limited computing power of the MCU that the firmware runs on may also impede the deployment of traffic analysis defenses.

6.3.1. Host Fuzzing and Analysis. Like firmware fuzzing, the fuzz input for the host stack can also come from a real device OTA [89], [102], [134] or from an emulated device [11], [103]. Similarly, fuzzing OTA provides better scalability but has lower performance. It is noteworthy that existing fuzzing works are either protocol-specific (e.g., SweynTooth [89] only supports BLE) or platform-specific (e.g., Frankenstein [11] only supports Linux while ToothPicker [103] only supports iOS). A generic fuzzing tool for the host layer is still missing. In terms of host code analysis, instead of analyzing the host code itself, a misconfigured device can be identified by analyzing its communicating counterpart (e.g., a device's companion app) [140], which is different from the firmware analysis.

6.3.2. Traffic Analysis and Filtering. Like other protocols, the traffic analysis can be performed at the network gateway [147] or a communication endpoint [148]. Bluetooth traffic can also be analyzed on a "gateway" device that sniffs the Bluetooth

TABLE 4: Comparison between attacks and defenses at different layers. Rows represent the defense techniques and columns show the attack types. Each cell indicates the comparison between the defense technique and the attack type. Complete prevention, partial prevention, complete detection, and partial detection are represented by ■, ◧, ●, and ◐ respectively. – means the defense cannot defend against the attack.

	Signal Eavesdropping	Signal Injection	Firmware Corruption	Breaking Key Sharing	Authentication Bypass	Breaking Encryption	Host Corruption	Illegal Service Access	Privacy Violation
	Physical Layer		Firmware Layer				Host Layer		
Device Identification	◧	◐	–	–	–	–	–	◐	◧
Firmware Fuzzing and Analysis	–	–	◐	◐	–	–	–	–	◐
Formal Verification	–	–	–	◐	–	–	–	–	–
Protocol Enhancement	◧	–	–	–	–	–	–	–	–
Host Fuzzing and Analysis	–	–	–	–	–	–	◐	–	◐
Traffic Analysis and Filtering	–	–	–	–	–	–	◧	◧	–
System Hardening	–	–	–	–	–	–	–	◧	–

traffic from around devices, or on an endpoint device that communicates with other devices.

On the one hand, analyzing the traffic on a "gateway" device [142] does not require modifications on the devices to be protected, but it needs extra hardware and faces the challenges introduced by Bluetooth frequency hopping [47], [55], [77]. On the other hand, while analyzing and filtering Bluetooth traffic on an endpoint [9], [143] does not need extra hardware and is not affected by frequency hopping, it requires software modifications on the protected device. As such, performing traffic analysis on an endpoint may not be applicable to low-end devices whose software cannot be modified.

6.3.3. System Hardening. Defenses in this subcategory focus on improving Bluetooth security by enhancing its implementations. The defenses either add extra code (e.g., access control for available services [105], [107] and additional security mechanisms, such as encryption and authentication at the application level [109]) or remove unused code [129] to harden the host code. Compared to the protocol enhancements (Section 6.2.3), hardening the host code does not change the protocol, and thus introduces little compatibility issues. For example, introducing extra access control policies [105], [107] or removing unused code [129] only need to modify the high-end device (e.g., Android phones) and does not affect the benign connections. Adding additional security mechanisms at the application level [109] requires modifications on both ends. However, since a device and its communicating counterpart (e.g., a companion app) are usually developed by the same vendor, the vendor can update the device and the app at the same time to minimize compatibility issues.

7. Defenses Takeaways

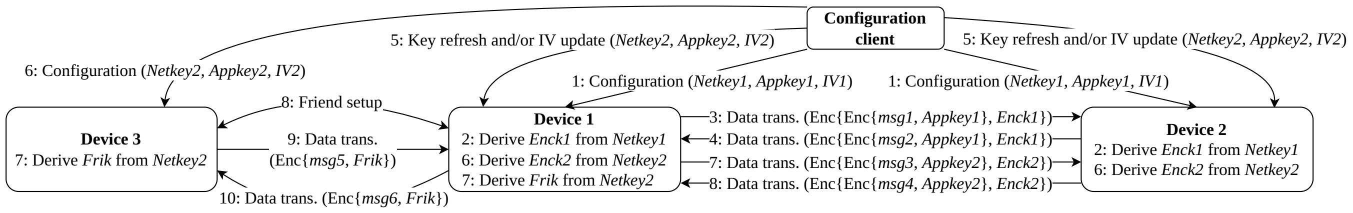
Table 3 summarizes the defenses we surveyed. By cross-checking the attacks' and defenses' target protocols, phases, and attack models, we highlight the relationships between the attack types and defense techniques in Table 4. In this table, we evaluate a defense technique as complete prevention (■) if it can completely mitigate a certain type of attack. Partial prevention (◧) represents that the considered defense works in some cases, but it cannot completely mitigate the attack. Complete detection (●) means that the defense technique can detect the attack with low false positives. Note that complete detection does not mean the defense works perfectly, since many of these solutions work only under specific assumptions. Finally, partial detection (◐) indicates that the defense can detect some attacks but not all attacks within a specific attack type. From the systematization of the studied defenses, we have the following takeaways:

T4. Bluetooth fuzzing tools are effective yet not comprehensive. Due to the stateful nature of Bluetooth, stateful fuzzing is effective to discover implementation bugs at both the firmware layer [89], [90] and the host layer [89], [102], [141]. However, compared to the firmware layer, which is well-supported [89], [90], existing tools have limited support for the host layer. For example, SweynTooth [89] can only fuzz the central role of BLE at the host layer. L2Fuzz [141] only fuzzes the L2CAP protocol without considering other protocols (e.g., SDP [58, p.1170]) and profiles (e.g., A2DP [149]). Because the state machines used to guide fuzzing in these tools are built manually and protocol-specific, existing tools cannot be readily adopted to fuzz other host protocols and profiles. A fuzzing tool that can fuzz BC and the peripheral role of BLE at the host layer (e.g., the BC and BLE implementation on Linux and Android) is still missing.

T5. There is no effective defense against attacks exploiting users' mistakes. While there are attacks exploiting user mistakes [94]–[96], no defense considers users' mistakes in its attack model. Accordingly, none of the existing defenses are effective against these attacks. To avoid compatibility issues (as discussed in **T2**), an effective approach to defend against such attacks might be improving the design of the user interfaces (UI) involved in pairing, as we will further discuss in Section 9.

T6. Defenses against authentication bypass and breaking encryption are still missing. As Table 4 shows (in the "authentication bypass" and "breaking encryption" columns), no existing defense techniques can defend against the authentication bypass and breaking encryption attacks. As a result, the only approach to protect a device from these attacks is to update its software and/or hardware to make it compatible with a newer version of the Bluetooth specification (5.2 and above). However, updating software is already challenging [150]–[152], not to mention hardware updates. As such, most of the devices that are vulnerable to these attacks, such as KNOB [5] and BIAS [7], are still affected by these attacks.

T7. Formal analysis of Bluetooth is incomplete. Formal analysis of Bluetooth is effective [8], [13]–[15] in finding design vulnerabilities in the specification. However, compared with BC


Configuration client
5: Key refresh and/or IV update (Netkey2, Appkey2, IV2)
5: Key refresh and/or IV update (Netkey2, Appkey2, IV2)
6: Configuration (Netkey2, Appkey2, IV2)
1: Configuration (Netkey1, Appkey1, IV1)
1: Configuration (Netkey1, Appkey1, IV1)
8: Friend setup
Device 3
7: Derive Frik from Netkey2
9: Data trans. (Enc{msg5, Frik})
10: Data trans. (Enc{msg6, Frik})
Device 1
2: Derive Enck1 from Netkey1
6: Derive Enck2 from Netkey2
7: Derive Frik from Netkey2
3: Data trans. (Enc{Enc{msg1, Appkey1}, Enck1})
4: Data trans. (Enc{Enc{msg2, Appkey1}, Enck1})
7: Data trans. (Enc{Enc{msg3, Appkey2}, Enck2})
8: Data trans. (Enc{Enc{msg4, Appkey2}, Enck2})
Device 2
2: Derive Enck1 from Netkey1
6: Derive Enck2 from Netkey2
Figure 2: Interactions of security-related protocols in Mesh. Steps with the same number can be executed independently.

and BLE, for which the security protocols (i.e., pairing, authentication, and encryption) have been fully modeled and analyzed, the existing formal analysis of Mesh [13] only considers its provisioning protocol and encryption. Other security-related protocols (e.g., the key refresh protocol) are not analyzed yet. To close this gap, as will be presented in Section 8, we provide a comprehensive formal analysis of Mesh, which also includes the key refresh, initial vector update, and friendship protocols.

8. Comprehensive Formal Analysis of Mesh

As discussed in **T7**, the existing formal analysis of Mesh does not support all security-related protocols. To close this gap, in this section, we present the design, implementation, and evaluation of our comprehensive Mesh formal model that covers all security-relate protocols. Note that the difference between the existing research [13] and our model is that the existing research does not cover the three protocols discussed in Section 8.1 (i.e., key refresh, initial vector update, and friendship) while our model supports these protocols. In our model, we assume the attacker has the capabilities defined in the Dolev-Yao attack model.

8.1. Model Design and Implementation

Before a Mesh device joins a Mesh network, it has to go through a configuration procedure (configured by the configuration client in Figure 2) in the key sharing phase, during which it receives a network key, an application key, and an initial vector (IV). After joining the Mesh network, in the data exchange phase, other security-related protocols in Mesh, i.e., the key refresh, IV update, and Mesh friendship protocols may be executed. Figure 2 illustrates the interactions between these security-related protocols.

We implement our model using ProVerif [16]. To align with the Dolev-Yao attack model, the messages exchanged between devices are via an *open channel*. We follow a similar way to model the configuration as existing work [13] and detail how to model the other three protocols as follows. Our model is publicly available to promote future research [17].

Key refresh. After configuration, Mesh uses the key refresh protocol to update a device's network and application keys. By refreshing the keys, the old keys can be revoked. There are three steps to refresh a network or application key. 1) Distribution of new keys: the configuration client sends new keys (*Netkey2* and *Appkey2* of Step 5 in Figure 2) encrypted using a *device key* (derived during the provisioning of the configuration procedure) to the device whose keys need to be replaced. 2) Switching to the new keys: after receiving the new keys from the configuration client, a device can only send messages encrypted using the new keys (Step 7 in Figure 2), but it can still receive messages encrypted using the old keys. 3) Revoking old keys: after receiving a notification message from the configuration client, the device only sends and receives messages encrypted using the new keys (Step 8 in Figure 2). A simplified implementation of the configuration client is shown in Appendix B.

IV update. The IV value is used to derive the nonces for encryption at the network and application layers. Updating the IV value prevents the reuse of the same nonce. The IV update starts from a Mesh device, such as the configuration client, sending a message that indicates the IV update and contains the new IV value. After receiving such a message, a device can accept messages with both old and new IV values, but it can only send messages with the old IV value. Then, the configuration client sends a notification message indicating the end of the IV update. Once receiving the notification message, the device can still accept messages with both old and new IVs, but it can only send messages with the new IV value. Note that IVs are not considered confidential values in Mesh and are transmitted in plaintext. The simplified IV update procedure (for the sender side) is shown in Appendix B.

Friendship. The friendship protocol allows a general device and a low-power device to establish a friendship through which the general device can store messages for the low-power device when it sleeps and send the stored messages to the low-power device when it is awake. To establish a friendship, the low-power device first sends a friend request message followed by the general device responding with a friend offer message. Then, the two devices derive the same key (*Frik* in Figure 2) from the network key and their MAC addresses for the network-level encryption. When the low-power device sends messages to the general device or vice versa (Steps 9 and 10 in Figure 2), at the network level, the messages are encrypted using the key derived in the previous step (i.e., *Frik*). The simplified low-power device is implemented as shown in Appendix B.

8.2. Evaluation

We verify the same properties as the existing work [13] for the configuration procedure. These properties are whether the configuration client and the device correctly authenticate each other (P1 and P2), and whether the distributed keys (*Netkey1* in Figure 2) from the configuration client and the response from the device are confidential (P3 and P4). Besides, we also verify the confidentiality of the distributed cryptographic keys during the key refresh procedure, denoted by P5. Finally, since the encryption is designed to guarantee the confidentiality of the transmitted messages between devices, we verify the confidentiality property of transmitted messages (*msg1* - *msg6* in Figure 2), denoted by P6 - P11, respectively.

As discussed in the existing work [13], the Mesh configuration has 8 different modes for provisioning (combining 2 different public key exchange methods with 4 authentication methods). We verify all the properties (P1 - P11) in all the 8 modes denoted by M1 to M8, respectively. Moreover, we also verify these properties after applying the fix to known attacks following the adversaries from Bluetooth SIG [36], [37], [64]. We run our verification on a server with two Intel (R) Xeon (R) Gold 6258R CPUs and 1TB of memory. The detailed results are shown in Table 5 in Appendix A.

As the table shows, our model can discover existing attacks, including BlueMAN [13], BlueMirror [21], and the MitM attacks against the non-authenticated configuration procedure. The "Fix" column in Table 5 shows that all the properties hold after applying the fix suggested by Bluetooth SIG.

Although our model covers more security-related protocols, it has limitations. Since ProVerif does not support the complete modeling of the exclusive or (xor) operation [153, p.45], our model cannot discover the malleability issue described in the existing work [21]. In addition, our model does not support the execution of devices with unbounded sessions.

9. Future Directions

Based on our systematization, we realized that the current defenses cannot completely secure Bluetooth, as discussed in **T4**, **T5**, and **T6**. As a partial mitigation, we took one step toward completing Bluetooth defenses by formally verifying security-related protocols in Bluetooth Mesh (Section 8). However, that is far from enough. Here, we point out the possible future directions that could help improve Bluetooth security.

Privacy exploration of BLE and Mesh. Due to the use of BLE advertising in the contact tracing mechanism [154]–[156] in response to the COVID-19 [157] pandemic, BLE's privacy in the device discovery phase has drawn a lot of attention from researchers, as discussed in **T1**. However, BLE privacy in the key sharing and data exchange phases is less explored, especially after unpairing. For example, when a user pairs her phone with another device (e.g., a smart lock of an Airbnb apartment) and unpairs them later, it is not clear if this device can recognize the phone and further infer the location of the phone's user. Moreover, no existing research focuses on Mesh privacy. Therefore, more exploration of BLE privacy in the key sharing and data exchange phases as well as a complete analysis of Mesh privacy is necessary.

Cross-stack security and privacy evaluation. BC and BLE allow the secret key of BC to be derived from the secret key of BLE and vice versa for a better user experience (only one pairing is needed to pair both BC and BLE). This design choice leads to the CSIA [13] and BLURTooth [114] attacks. In terms of privacy, the privacy of a device's BLE stack can be compromised via its BC stack [73] because of their coexistence. Besides, BLE and Mesh also share the firmware layer, physical layers, and the GATT profile (for proxy devices) at the host layer, which may also have security and privacy implications. Therefore, more security and privacy evaluation of BT-BLE, BLE-Mesh dual-stack, or even triple-stack devices should be conducted.

Host layer fuzzing. As discussed in **T4**, existing tools have limitations in fuzzing the host layer. Accordingly, the implementation security of frequently used profiles (e.g., A2DP and HID [158]) at the host layer is not thoroughly studied. One research direction is to build fuzzing tools that support all protocols and profiles implemented at the host layer. Ideally, stateful fuzzing will be preferred, based on the effectiveness of existing stateful fuzzing tools at the firmware layer [89], [90]. A possible challenge is how to build a state machine to guide fuzzing across different protocols (e.g., L2CAP and SDP) given that every protocol has its own state machine.

Better UI design of Bluetooth pairing for security and its evaluation. **T2** shows that users' mistakes during pairing can lead to attacks, and no existing defense is effective against such attacks without introducing compatibility concerns, as discussed in **T5**. To prevent these attacks, one possible solution is to design a better UI for pairing to help users avoid making mistakes. In fact, the SIG suggests differentiating pairing methods in the UI to avoid users erroneously approving the pairing [95], [96], [98]. However, in contrast with other fields (e.g., browser security [159], [160]), no systematic study has been performed on the usability and security of the UIs used by different devices during pairing. As such, how to design a better UI for Bluetooth pairing and how to conduct user studies to properly evaluate the design are still open questions, which are worth exploring.

Minimizing the functionality of firmware. Though several vulnerabilities have been found in Bluetooth firmware, the closed-source nature and the absence of patching mechanisms limit the prevention and mitigation of these attacks. As a result, the devices are left vulnerable if no other mitigation is proposed, as discussed in **T6**. For instance, the pairing procedure of BLE is implemented in the host while the pairing procedure of BC is in the firmware. Therefore, developers can fix the KNOB [5], [6] vulnerability of BLE by modifying the host code. While for BC, developers can only implement workarounds at the host layer [161] since the firmware is closed source and usually cannot be modified. One possible direction to address this issue is moving some firmware functionalities to the host so that it is easier to patch vulnerabilities.

Deprecation of legacy pairing methods. Table 1 shows that legacy pairing methods are vulnerable to diverse attacks, and one solution is to use the SCO mode where the most advanced security features are used. However, as discussed in **T2**, recent attacks [95], [96] are still effective if one device is in the SCO mode and the other one is using a legacy pairing method. Defense against these attacks relies on users to verify that two pairing devices use the same (secure) pairing method (i.e., SSP for BC and SCP for BLE). Users may be able to verify the pairing method if both devices have interfaces allowing for both input and output (e.g., smartphones and laptops). For example, when pairing two smartphones, each phone can show the pairing method it uses on its screen. However, it is challenging for users to verify the method if one device has limited user interfaces for input or output (e.g., Bluetooth headsets and bulbs). One fundamental solution could be deprecating the legacy pairing methods and only supporting the secure ones. However, this solution needs proper evaluation of its cost and usability impact on end-users, and it necessarily needs close coordination and collaboration between the SIG and Bluetooth vendors.

10. Conclusion

To understand Bluetooth security as a research domain, we have analyzed 76 attacks and 33 defenses since Bluetooth was introduced, covering more than 20 years of Bluetooth security research. Our systematization provides new insights regarding Bluetooth attacks and defenses and suggests promising research directions. The formal verification of Mesh we provided takes one step towards securing Bluetooth Mesh. In summary, our work provides a foundation to secure King Harald's "legacy" in the future.

Acknowledgment

We thank the anonymous reviewers and shepherd for their valuable comments and suggestions. This work was supported in part by the National Science Foundation (NSF) under Grant CNS-2145744 and the Office of Naval Research (ONR) under Grants N00014-20-1-2128 and N00014-22-1-2671. Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of NSF or ONR.

References

[1] W. Mingren, "Bluetooth: Why Modern Tech is Named After Powerful King of Denmark and Norway," https://www.ancient-origins.net/history-famous-people/bluetooth-why-modern-tech-named-after-powerful-king-denmark-and-norway-007398, 2017, accessed: August 1, 2020. 1

[2] Bluetooth Special Interest Group, "2022 Bluetooth Market Update," https://www.bluetooth.com/2022-market-update/?utm_campaign=bmu%26utm_source=internal%26utm_medium=web%26utm_content=2022bmu-resourcepopup, 2022, accessed: April 15, 2022. 1

[3] Armis, "BlueBorne Technical White Paper," https://www.armis.com/research/blueborne/, 2020, accessed: January 29, 2020. 1, 8, 9

[4] ——, "BLEEDINGBIT," https://www.armis.com/bleedingbit/, 2020, accessed: July 26, 2020. 1, 7, 8

[5] D. Antonioli, N. O. Tippenhauer, and K. Rasmussen, "The KNOB is Broken: Exploiting Low Entropy in the Encryption Key Negotiation Of Bluetooth BR/EDR," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2019. 1, 3, 8, 12, 14

[6] ——, "Key Negotiation Downgrade Attacks on Bluetooth and Bluetooth Low Energy," *ACM Transactions on Privacy and Security*, vol. 23, no. 3, 2020. 1, 3, 4, 5, 8, 14

[7] ——, "BIAS: Bluetooth Impersonation AttackS," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2020. 1, 3, 7, 8, 12

[8] J. Wu, Y. Nan, V. Kumar, D. J. Tian, A. Bianchi, M. Payer, and D. Xu, "BLESA: Spoofing Attacks against Reconnections in Bluetooth Low Energy," in *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*, 2020. 1, 2, 8, 9, 10, 11, 12

[9] D. J. Tian, G. Hernandez, J. I. Choi, V. Frost, P. C. Johnson, and K. R. B. Butler, "LBM: A Security Framework for Peripherals within the Linux Kernel," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2019. 1, 11, 12

[10] D. Mantz, J. Classen, M. Schulz, and M. Hollick, "InternalBlue - Bluetooth Binary Patching and Experimentation Framework," in *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2019. 1, 7, 8, 10, 11

[11] J. Ruge, J. Classen, F. Gringoli, and M. Hollick, "Frankenstein: Advanced Wireless Fuzzing to Exploit New Bluetooth Escalation Targets," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2020. 1, 7, 8, 10, 11

[12] J. Wu, Y. Nan, V. Kumar, M. Payer, and D. Xu, "BlueShield: Detecting Spoofing Attacks in Bluetooth Low Energy (BLE) Networks," in *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2020. 1, 10, 11

[13] J. Wu, R. Wu, D. Xu, D. J. Tian, and A. Bianchi, "Formal Model-Driven Discovery of Bluetooth Protocol Design Vulnerabilities," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2022. 2, 3, 5, 7, 8, 9, 10, 11, 12, 13, 14, 19

[14] M. K. Jangid, Y. Zhang, and Z. Lin, "Extrapolating formal analysis to uncover attacks in bluetooth passkey entry pairing," 2023. 2, 10, 11, 12

[15] C. Cremers and D. Jackson, "Prime, Order Please! revisiting Small Subgroup and Invalid Curve Attacks on Protocols Using Diffie-Hellman," in *Proceedings of the IEEE Computer Security Foundations Symposium (CSF)*, 2019. 2, 3, 4, 7, 8, 10, 11, 12

[16] B. Blanchet, "Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif," *Foundations and Trends in Privacy and Security*, vol. 1, 2016. 2, 13

[17] "Code repo of our Mesh model," https://github.com/purseclab/mesh-model, 2023. 2, 13

[18] Y. Shaked and A. Wool, "Cracking the Bluetooth PIN," in *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2005. 2, 3, 7, 8

[19] M. Jakobsson and S. Wetzel, "Security Weaknesses in Bluetooth," in *Proceedings of the Conference on Topics in Cryptology: The Cryptographer's Track at RSA (CT-RSA)*, 2001. 2, 3, 7, 8

[20] D.-Z. Sun and X.-H. Li, "Vulnerability and Enhancement on Bluetooth Pairing and Link Key Generation Scheme for Security Modes 2 and 3 ," in *Proceedings of the International Conference on Information and Communications Security (ICICS)*, 2016. 2, 3, 7, 8

[21] T. Claverie and J. L. Esteves, "BlueMirror: Reflections on Bluetooth Pairing and Provisioning Protocols," in *Proceedings of the IEEE Security and Privacy Workshops (SPW)*, 2021. 2, 3, 4, 5, 7, 8, 14, 19

[22] J. Barnickel, J. Wang, and U. Meyer, "Implementing an Attack on Bluetooth 2.1+ Secure Simple Pairing in Passkey Entry Mode," in *Proceedings of the IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2012. 2, 3, 4, 7, 8

[23] D.-Z. Sun, Y. Mu, and W. Susilo, "Man-in-the-Middle Attacks on Secure Simple Pairing in Bluetooth Standard V5.0 and Its Countermeasure," *Personal Ubiquitous Computing*, vol. 22, no. 1, 2018. 2, 3, 4, 7, 8

[24] A. Y. Lindell, "Attacks on the Pairing Protocol of Bluetooth v2.1," *Black Hat USA*, 2008. 2, 3, 4, 7, 8

[25] E. Biham and L. Neumann, "Breaking the Bluetooth Pairing–The Fixed Coordinate Invalid Curve Attack," in *Proceedings of the International Conference on Selected Areas in Cryptography (SAC)*, 2019. 2, 3, 4, 7, 8

[26] Bluetooth Special Interest Group, "Bluetooth SIG Statement Regarding the Bluetooth Impersonation Attacks (BIAS) Security Vulnerability," https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/bias-vulnerability/, 2021, accessed: July 1, 2022. 3

[27] M. Hermelin and K. Nyberg, "Correlation properties of the bluetooth combiner," in *Proceedings of the International Conference on Information Security and Cryptology (ICISC)*, 1999. 3, 4, 5, 8

[28] Y. Lu and S. Vaudenay, "Faster Correlation Attack on Bluetooth Keystream Generator E0," in *Proceedings of the International Cryptology Conference (CRYPTO)*, 2004. 3, 4, 5, 8

[29] S. Fluhrer and S. Lucks, "Analysis of the E0 Encryption System," in *Proceedings of the International Conference on Selected Areas in Cryptography (SAC)*, 2001. 3, 4, 5, 8

[30] Y. Lu, W. Meier, and S. Vaudenay, "The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption," in *Proceedings of the International Cryptology Conference (CRYPTO)*, 2005. 3, 4, 5, 8

[31] Y. Lu and S. Vaudenay, "Cryptanalysis of Bluetooth Keystream Generator Two-Level E0," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt)*, 2004. 3, 4, 5, 8

[32] M. Ryan, "Bluetooth: With Low Energy Comes Low Security," in *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*, 2013. 3, 4, 7, 8

[33] J. K. Becker, D. Li, and D. Starobinski, "Tracking Anonymized Bluetooth Devices," *Proceedings of Privacy Enhancing Technologies*, vol. 2019, no. 3, 2019. 3, 4, 6, 7, 8, 9

[34] K. Fawaz, K.-H. Kim, and K. G. Shin, "Protecting Privacy of BLE Device Users," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2016. 3, 4, 8, 9, 10, 11

[35] Bluetooth Special Interest Group, "Bluetooth SIG Statement Regarding the 'Predictable AuthValue in Bluetooth Mesh Provisioning Leads to MITM' Vulnerability," https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/predicatable-authvalue/, 2021, accessed: July 1, 2022. 3

[36] ——, "Bluetooth SIG Statement Regarding the 'Malleable Commitment' Vulnerability," https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/malleable/, 2021, accessed: July 1, 2022. 3, 5, 14

[37] ——, "Bluetooth SIG Statement Regarding the 'Impersonation Attack in Bluetooth Mesh Provisioning' Vulnerability," https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/impersonation-mesh/, 2021, accessed: August 1, 2021. 3, 5, 14

[38] ——, "Bluetooth Core Specifications 1.0B," 1999. 2

[39] ——, "Bluetooth Core Specifications v2.0 + EDR," 2005. 2

[40] ——, "Bluetooth Core Specifications 3.0 + HS," 2009. 2

[41] ——, "Bluetooth Core Specifications 5.3," 2021. 2, 3, 4, 5, 7

[42] ——, "Bluetooth Core Specifications v2.1 + EDR," 2007. 2

[43] K. Haataja and P. Toivanen, "Two Practical Man-In-The-Middle Attacks on Bluetooth Secure Simple Pairing and Countermeasures," *IEEE Transactions on Wireless Communications*, vol. 9, no. 1, 2010. 2, 7, 8

[44] K. Hypponen and K. M. Haataja, ""Nino" man-in-the-middle attack on bluetooth secure simple pairing," in *Proceedings of the IEEE/IFIP International Conference in Central Asia on Internet (ICI)*, 2007. 2, 7, 8

[45] K. M. Haataja and K. Hypponen, "Man-In-The-Middle Attacks on Bluetooth: a Comparative Analysis, a Novel Attack, and Countermeasures," in *Proceedings of the International Symposium on Communications, Control and Signal Processing (ISCCSP)*, 2008. 2, 7, 8

[46] T. R. Mutchukota, S. K. Panigrahy, and S. K. Jena, "Man-In-The-Middle Attack and Its Countermeasure in Bluetooth Secure Simple Pairing," in *Proceedings of the International Conference on Information Processing (ICIP)*, 2011. 2, 7, 8

[47] M. A. Albahar, K. Haataja, and P. Toivanen, "Bluetooth MITM Vulnerabilities: A Literature Review, Novel Attack Scenarios, Novel Countermeasures, and Lessons Learned," *International Journal on Information Technologies and Security*, vol. 8, no. 4, 2016. 2, 7, 8, 12

[48] K. Haataja and P. Toivanen, "Practical Man-in-the-Middle Attacks Against Bluetooth Secure Simple Pairing," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCom)*, 2008. 2, 7, 8

[49] Bluetooth Special Interest Group, "Bluetooth Core Specifications 5.1," 2019. 2, 4

[50] ——, "Bluetooth Core Specifications 4.0," 2010. 3, 4

[51] C. Fontaine, *E0 (Bluetooth)*, 2005, pp. 169–169. 3

[52] Bluetooth Special Interest Group, "Bluetooth Core Specifications 4.1," 2013. 3

[53] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality," https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf, 2004, accessed: August 1, 2020. 3

[54] Bluetooth Special Interest Group, "Bluetooth Core Specifications 5.2," 2019. 3, 4

[55] 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 *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2020. 3, 6, 8, 12

[56] National Institute of Standards and Technology, "Annex A:Approved Security Functions for FIPS PUB 140-2, Security Requirements for Cryptographic Modules," https://csrc.nist.gov/csrc/media/publications/fips/140/2/final/documents/fips1402annexa.pdf, 2019, accessed: August 1, 2020. 3

[57] Bluetooth Special Interest Group, "Bluetooth Core Specifications 4.2," 2014. 4

[58] ——, "Bluetooth Core Specifications 5.4," 2023. 4, 7, 12

[59] ——, "Bluetooth Core Specifications 5.0," 2016. 4

[60] 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 International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2016. 4, 6, 7, 8

[61] G. Celosia and M. Cunche, "Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols," *Proceedings on Privacy Enhancing Technologies*, 2020. 4

[62] 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 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022. 4, 7, 8

[63] Bluetooth Special Interest Group, "Mesh Profile Specification 1.0.1," https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=457092, 2019, accessed: June 10, 2020. 4, 5, 7, 8

[64] ——, "Bluetooth SIG Statement Regarding the 'AuthValue Leak' Vulnerability," https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/authvalue-leak/, 2021, accessed: July 1, 2022. 5, 14

[65] ——, "Traditional Profile Specifications," https://www.bluetooth.com/specifications/profiles-overview/, 2019, accessed: August 1, 2019. 5

[66] M. Weller, J. Classen, F. Ullrich, D. Waßmann, and E. Tews, "Lost and Found: Stopping Bluetooth Finders from Leaking Private Information," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2020. 6

[67] The Hacker News, "The Hacker News," https://thehackernews.com/, 2023. 6

[68] FeedSpot, "Top 50 Cyber Security News Websites for Information Security Pros," https://blog.feedspot.com/cyber_security_news_websites/, 2023. 6

[69] D. Dolev and A. Yao, "On the Security of Public Key Protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, 1983. 6

[70] Ettus Research, "USRP B210 (Board Only)," https://www.ettus.com/all-products/ub210-kit/, 2023. 6, 8

[71] Ubertooth Developers, "Ubertooth One," https://github.com/greatscottgadgets/ubertooth/wiki, 2019, accessed: August 1, 2019. 6, 8

[72] Amazon, "Long Range USB Bluetooth 5.1 Adapter for PC USB Bluetooth Adapter Wireless Audio Dongle 328FT/100M 5.1 Bluetooth Transmitter Receiver for Desktop Laptop PC with Windows 11/10/8/8.1/7," https://www.amazon.com/Bluetooth-Adapter-Wireless-Transmitter-Receiver/dp/B09KG7QQ5V/ref=sr_1_5?keywords=bluetooth+dongleŽqid=1679078149Žsr=8-5, 2023. 6, 8

[73] N. Ludant, T. D. Vo-Huu, S. Narain, and G. Noubir, "Linking Bluetooth LE & Classic and Implications for Privacy-Preserving Bluetooth-Based Protocols," in *Proceeding of the IEEE Symposium on Security and Privacy (S&P)*, 2021. 6, 7, 8, 14

[74] H. Givehchian, N. Bhaskar, E. R. Herrera, H. R. L. Soto, C. Dameff, D. Bharadia, and A. Schulman, "Evaluating Physical-Layer BLE Location Tracking Attacks on Mobile Devices," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2022. 6, 7, 8

[75] R. Cayre, F. Galtier, G. Auriol, V. Nicomette, M. Kaâniche, and G. Marconato, "InjectaBLE: Injecting malicious traffic into established Bluetooth Low Energy connections," in *Proceedings of Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021. 6, 7, 8

[76] T. Tucker, H. Searle, K. Butler, and P. Traynor, "Blue's Clues: Practical Discovery of Non-Discoverable Bluetooth Devices," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2023. 6, 7, 8

[77] D. Spill and A. Bittau, "BlueSniff: Eve Meets Alice and Bluetooth," in *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*, 2007. 6, 8, 12

[78] W. Albazrqaoe, J. Huang, and G. Xing, "Practical Bluetooth Traffic Sniffing: Systems and Privacy Implications," in *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2016. 6, 7, 8

[79] P. Staat, K. Jansen, C. Zenger, H. Elders-Boll, and C. Paar, "Analog Physical-Layer Relay Attacks with Application to Bluetooth and Phase-Based Ranging," in *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2022. 6, 8

[80] H. Zhang, A. K. Maji, and S. Bagchi, "Privacy in the Mobile World: An Analysis of Bluetooth Scan Traces," in *Proceedings of the Joint Workshop on CPS and IoT Security and Privacy (CPSIOTSEC)*, 2020. 6, 7, 8, 9

[81] Apple, "Find My," https://www.apple.com/icloud/find-my/, 2022, accessed: January 10, 2022. 6

[82] ——, "Use Continuity to connect your Mac, iPhone, iPad, iPod touch, and Apple Watch," https://support.apple.com/en-us/HT204681, 2022, accessed: January 10, 2022. 6

[83] A. Heinrich, M. Stute, T. Kornhuber, and M. Hollick, "Who Can Find My Devices? Security and Privacy of Apple's Crowd-Sourced Bluetooth Location Tracking System," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 3, 2021. 7, 8

[84] 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 *Proceeding of the USENIX Security Symposium (USENIX Security)*, 2021. 7, 8

[85] J. Martin, D. Alpuche, K. Bodeman, L. Brown, E. Fenske, L. Foppe, T. Mayberry, E. C. 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*, 2019. 7, 8

[86] A. Mariotto, A. Heinrich, D. Kreitschmann, G. Noubir, and M. Hollick, "A Billion Open Houses for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2019. 7, 8

[87] D. Cross, J. Hoeckle, M. Lavine, J. Rubin, and K. Snow, "Detecting Non-Discoverable Bluetooth Devices," in *Proceedings of the International Conference on Critical Infrastructure Protection (ICCIP)*, 2007. 7, 8

[88] D. Cauquil, "Defeating Bluetooth Low Energy 5 PRNG for Fun and Jamming," *DEF CON*, 2019. 7, 8

[89] M. E. Garbelini, C. Wang, S. Chattopadhyay, S. Sumei, and E. Kurniawan, "SweynTooth: Unleashing Mayhem over Bluetooth Low Energy," in *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)*, 2020. 7, 8, 9, 10, 11, 12, 14

[90] M. E. Garbelini, S. Chattopadhyay, V. Bedi, S. Sun, and E. Kurniawan, "BrakTooth: Causing Havoc on Bluetooth Link Manager via Directed Fuzzing," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2022. 7, 8, 10, 11, 12, 14

[91] J. Classen, F. Gringoli, M. Hermann, and M. Hollick, "Attacks on Wireless Coexistence: Exploiting Cross-Technology Performance Features for Inter-Chip Privilege Escalation," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2022. 7, 8

[92] V. Kovah, "Finding New Bluetooth Low Energy Exploits via Reverse Engineering Multiple Vendors' Firmwares," *DEF CON*, 2020. 7, 8

[93] J. Classen, A. Heinrich, R. Reith, and M. Hollick, "Evil Never Sleeps: When Wireless Malware Stays On After Turning Off iPhones," in *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2022. 7, 8

[94] M. von Tschirschnitz, L. Peuckert, F. Franzen, and J. Grossklags, "Method Confusion Attack on Bluetooth Pairing," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2021. 7, 8, 9, 10, 12

[95] ANSSI, "Pairing Mode Confusion in BLE Passkey Entry," https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/confusion-in-ble-passkey/, 2022. 7, 8, 9, 12, 14

[96] ——, "Pairing Mode Confusion in BR/EDR," https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/confusion-in-br-edr/, 2022. 7, 8, 9, 12, 14

[97] J. Classen and M. Hollick, "Happy MitM: fun and toys in every Bluetooth device," in *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2021. 7, 8

[98] Bluetooth Special Interest Group, "Bluetooth SIG Statement Regarding the Method-Confusion Pairing Vulnerability," https://www.bluetooth.com/learn-about-bluetooth/key-attributes/bluetooth-security/method-vulnerability/, 2020, accessed: August 1, 2021. 7, 14

[99] A. Levi, E. Çetintaş, M. Aydos, Ç. K. Koç, and M. U. Ça ğlayan, "Relay Attacks on Bluetooth Authentication and Solutions," in *Proceedings of the International Symposium on Computer and Information Sciences (ISCIS)*, 2004. 7, 8

[100] M. Beccaro and M. Collura, "Extracting the painful (blue)tooth," *DEF CON*, 2015. 8

[101] J. Ruge, "CVE-2020-0022 an Android 8.0-9.0 Bluetooth Zero-Click RCE – BlueFrag," https://insinuator.net/2020/04/cve-2020-0022-an-android-8-0-9-0-bluetooth-zero-click-rce-bluefrag/, 2020, accessed: August 2, 2020. 8

[102] H. Yan, L. Qu, and D. Ke, "BrokenMesh: New Attack Surfaces of Bluetooth Mesh," *DEF CON*, 2022. 8, 9, 11, 12

[103] D. Heinze, J. Classen, and M. Hollick, "ToothPicker: Apple Picking in the iOS Bluetooth Stack," in *Proceedings of the USENIX Workshop on Offensive Technologies (WOOT)*, 2020. 8, 11

[104] G. S. Research, "Linux: Heap-Based Type Confusion in L2CAP (BleedingTooth)," https://google.github.io/security-research/pocs/linux/bleedingtooth/writeup.html, accessed: October 20, 2020. 8, 9

[105] M. Naveed, X.-y. Zhou, S. Demetriou, X. Wang, and C. A. Gunter, "Inside Job: Understanding and Mitigating the Threat of External Device Mis-Binding on Android," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014. 8, 9, 11, 12

[106] C. Koh, J. Kwon, and J. Hur, "BLAP: Bluetooth Link Key Extraction and Page Blocking Attacks," in *Proceedings of the Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022. 8, 9

[107] F. Xu, W. Diao, Z. Li, J. Chen, and K. Zhang, "BadBluetooth: Breaking Android Security Mechanisms via Malicious Bluetooth Peripherals," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2019. 8, 9, 11, 12

[108] M. Ai, K. Xue, B. Luo, L. Chen, N. Yu, Q. Sun, and F. Wu, "Blacktooth: Breaking through the Defense of Bluetooth in Silence," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022. 8, 9

[109] P. Sivakumaran and J. Blasco, "A Study of the Feasibility of Co-located App Attacks against BLE and a Large-Scale Analysis of the Current Application-Layer Security Landscape," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2019. 8, 9, 11, 12

[110] J. Wang, F. Hu, Y. Zhou, Y. Liu, H. Zhang, and Z. Liu, "BlueDoor: Breaking the Secure Information Flow via BLE Vulnerability," in *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2020. 8, 9

[111] Y. Zhang, J. Weng, R. Dey, Y. Jin, Z. Lin, and X. Fu, "Breaking Secure Pairing of Bluetooth Low Energy Using Downgrade Attacks," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2020. 8, 9

[112] S. Jasek, "Gattacking Bluetooth smart devices," in *Black Hat USA*, 2016. 8

[113] F. Álvarez, L. Almon, A.-S. Hahn, and M. Hollick, "Toxic Friends in Your Network: Breaking the Bluetooth Mesh Friendship Concept," in *Proceedings of the ACM Workshop on Security Standardisation Research Workshop (SSR)*, 2019. 8

[114] D. Antonioli, N. O. Tippenhauer, K. Rasmussen, and M. Payer, "BLURTooth: Exploiting Cross-Transport Key Derivation in Bluetooth Classic and Bluetooth Low Energy," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS)*, 2022. 8, 9, 14

[115] P. Kitsos, N. Sklavos, K. Papadomanolakis, and O. Koufopavlou, "Hardware Implementation of Bluetooth Security," *IEEE Pervasive Computing*, vol. 2, no. 1, 2003. 8

[116] M. D. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, and C. E. Goutis, "Comparison of the Hardware Implementation of Stream Ciphers," *International Arab Journal of Information Technology*, vol. 2, no. 4, 2005. 8

[117] BlueZ contributors, "BlueZ," http://www.bluez.org/, 2019, accessed: August 1, 2019. 8

[118] Linux contributors, "Kernel Part of BlueZ," https://github.com/torvalds/linux/tree/master/net/bluetooth, 2022, accessed: June 30, 2022. 9

[119] R. Mayrhofer, J. V. Stoep, C. Brubaker, and N. Kralevich, "The Android Platform Security Model," *ACM Transactions on Privacy and Security*, vol. 24, no. 3, 2021. 9

[120] J. Huang, W. Albazrqaoe, and G. Xing, "BlueID: A Practical System for Bluetooth Device Identification," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2014. 10, 11

[121] J. Symon, "Detecting Relay Attacks against Bluetooth Communications on Android," Ph.D. dissertation, The University of Waikato, 2018. 10, 11

[122] B. Feng, A. Mera, and L. Lu, "P2IM: Scalable and Hardware-independent Firmware Testing via Automatic Peripheral Interface Modeling," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2020. 10

[123] Y. Zheng, A. Davanian, H. Yin, C. Song, H. Zhu, and L. Sun, "FIRM-AFL: High-Throughput Greybox Fuzzing of IoT Firmware via Augmented Process Emulation," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2019. 10

[124] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A Large-Scale Analysis of the Security of Embedded Firmwares," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2014. 10

[125] D. D. Chen, M. Woo, D. Brumley, and M. Egele, "Towards Automated Dynamic Analysis for Linux-based Embedded Firmware," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2016. 10

[126] J. Classen and M. Hollick, "Inside Job: Diagnosing Bluetooth Lower Layers Using off-the-Shelf Devices," in *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2019. 10, 11

[127] F. Fowze, D. J. Tian, G. Hernandez, K. Butler, and T. Yavuz, "ProXray: Protocol Model Learning and Guided Firmware Analysis," *IEEE Transactions on Software Engineering*, 2019. 10, 11

[128] H. Wen, Z. Lin, and Y. Zhang, "FirmXRay: Detecting Bluetooth Link Layer Vulnerabilities from Bare-Metal Firmware," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020. 10, 11

[129] J. Wu, R. Wu, D. Antonioli, M. Payer, N. O. Tippenhauer, D. Xu, D. J. Tian, and A. Bianchi, "LIGHTBLUE: Automatic Profile-Aware Debloating of Bluetooth Stacks," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2021. 10, 11, 12

[130] B. Blanchet, "Automatic Proof of Strong Secrecy for Security Protocols," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2004. 10

[131] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN Prover for the Symbolic Analysis of Security Protocols," in *Proceedings of the International Conference on Computer Aided Verification (CAV)*, 2013. 10

[132] R. Chang and V. Shmatikov, "Formal Analysis of Authentication in Bluetooth Device Pairing," *Proceedings of the LICS/ICALP Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA)*, 2007. 10, 11

[133] M. Shi, J. Chen, K. He, H. Zhao, M. Jia, and R. Du, "Formal Analysis and Patching of BLE-SC Pairing," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2023. 10, 11

[134] I. Karim, A. Al Ishtiaq, S. R. Hussain, and E. Bertino, "BLEDiff: Scalable and Property-Agnostic Noncompliance Checking for BLE Implementations," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2023. 11

[135] D.-Z. Sun and L. Sun, "On Secure Simple Pairing in Bluetooth Standard v5. 0-Part I: Authenticated Link Key Security and Its Home Automation and Entertainment Applications," *Sensors*, vol. 19, no. 5, 2019. 11

[136] K. Arai and T. Kaneko, "Formal Verification of Improved Numeric Comparison Protocol for Secure Simple Pairing in Bluetooth Using ProVerif," in *Proceedings of the International Conference on Security and Management (SAM)*, 2014. 11

[137] P. Wang, "Bluetooth Low Energy-privacy enhancement for advertisement," Master's thesis, Norwegian University of Science and Technology, 2014. 11

[138] A. Hassidim, Y. Matias, M. Yung, and A. Ziv, "Ephemeral Identifiers: Mitigating Tracking & Spoofing Threats to BLE Beacons," https://developers.google.com/beacons/eddystone-eid-preprint.pdf, 2016, accessed: August 1, 2019. 11

[139] D. Heinze, J. Classen, and F. Rohrbach, "MagicPairing: Apple's Take on Securing Bluetooth Peripherals," in *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2020. 11

[140] 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 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019. 11

[141] H. Park, C. K. Nkuba, S. Woo, and H. Lee, "L2Fuzz: Discovering Bluetooth L2CAP Vulnerabilities Using Stateful Fuzz Testing," in *Proceedings of Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022. 11, 12

[142] T. OConnor and D. Reeves, "Bluetooth Network-Based Misuse Detection," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2008. 11, 12

[143] T. Peters, T. J. Pierson, S. Sen, J. Camacho, and D. Kotz, "Recurring Verification of Interaction Authenticity Within Bluetooth Networks," in *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2021. 11, 12

[144] H. Krawczyk, *Perfect Forward Secrecy*, 2005, pp. 457–458. 11

[145] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network Intrusion Detection for IoT Security Based on Learning Techniques," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, 2019. 11

[146] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2017. 11

[147] P. A. Porras and A. Valdes, "Live Traffic Analysis of TCP/IP Gateways," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 1998. 11

[148] A. W. Moore and K. Papagiannaki, “Toward the Accurate Identification of Network Applications,” in *Proceedings of International Workshop on Passive and Active Network Measurement*, 2005. 11

[149] Bluetooth Special Interest Group, “Advanced Audio Distribution v1.3.2,” https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=457083, 2019, accessed: January 17, 2020. 12

[150] F. Paul, “Fixing, upgrading and patching IoT devices can be a real nightmare,” https://www.networkworld.com/article/3222651/fixing-upgrading-and-patching-iot-devices-can-be-a-real-nightmare.html, 2017, accessed: August 1, 2019. 12

[151] J. Borgini, “6 common challenges to IoT OTA updates,” https://www.techtarget.com/iotagenda/tip/6-common-challenges-to-IoT-OTA-updates, 2020, accessed: July 16, 2022. 12

[152] X. Zou, “IoT devices are hard to patch: Here’s why—and how to deal with security,” https://techbeacon.com/security/iot-devices-are-hard-patch-heres-why-how-deal-security, 2017, accessed: July 16, 2022. 12

[153] B. Blanchet, B. Smyth, V. Cheval, and M. Sylvestre, “ProVerif 2.04: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial,” https://bblanche.gitlabpages.inria.fr/proverif/manual.pdf, 2021, accessed: March 1, 2022. 14

[154] H. Cho, D. Ippolito, and Y. W. Yu, “Contact Tracing Mobile Apps for COVID-19: Privacy Considerations and Related Trade-offs,” 2020. 14

[155] P. C. Ng, P. Spachos, and K. Plataniotis, “COVID-19 and Your Smartphone: BLE-based Smart Contact Tracing,” 2020. 14

[156] Google, “Exposure Notifications: Using technology to help public health authorities fight COVID-19,” https://www.google.com/covid19/exposurenotifications/, 2020, accessed: November 1, 2020. 14

[157] C. for Disease Control and Prevention, “Coronavirus (COVID-19),” https://www.cdc.gov/coronavirus/2019-ncov/index.html, 2020, accessed: November 1, 2020. 14

[158] Bluetooth Special Interest Group, “Human Interface Device Profile,” https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=309012, 2020, accessed: August 1, 2020. 14

[159] J. Weinberger and A. P. Felt, “A Week to Remember: The Impact of Browser Warning Storage Policies,” in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2016. 14

[160] C. Thompson, M. Shelton, E. Stark, M. Walker, E. Schechter, and A. P. Felt, “The Web’s Identity Crisis: Understanding the Effectiveness of Website Identity Indicators,” in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2019. 14

[161] Archie Pusaka, “[PATCH v3] Bluetooth: Check for encryption key size on connect,” https://lore.kernel.org/linux-bluetooth/20200922155548.v3.1.I67a8b8cd4def8166970ca37109db46d731b62bb6Žchangeid/, 2020. 14

Appendix A. Detailed Result of Mesh Verification (Table 5)

Appendix B. Simplified Model Implementation in ProVerif

```
//Simplified implementation of key refresh
//(the configuration client)
let key_refresh_send() = (...//Send new keys
  let kencd = AES_CCM(newk, devkey, devn) in
  let nkencd = AES_CCM(kencd, enck, net_n) in
  let pecb = e(pri_key, concat(iv, nkencd)) in
  out(ch, (obfuscate(net_n, pecb), nkencd));
  //Send notification
  let net_n2 = net_n(ttl1, seq2, addr_p, iv) in
  let devn2 = dev_n(seq2, addr_p, addr, iv) in
  let nencd = AES_CCM(note1, devkey, devn2) in
  let notencd = AES_CCM(nencd, enck, net_n2) in
  let pecb = e(pri_key, concat(iv, notencd)) in
  out(ch, (obfuscate(net_n2, pecb), notencd))).
```

```
//Simplified IV update procedure (the sender side)
let iv_update_send() = (//Send new IV
  get mesh_net_key_c(addr, netkey) in
  let bkey = k1(netkey, s1(nkbk), P) in
  let mac = AES_CMAC(bkey, newiv) in
  out(ch, (newiv, mac));
  //Send notification
  let mac2 = AES_CMAC(bkey, note2) in
  out(ch, (note2, mac2))).
```

```
//Simplified friendship establishment
//implementation (low power device)
let friend_request() = (... //Send friend request
  let enck= enck(k2(netkey, ZERO)) in
  let pri_key = prik(k2(netkey, ZERO)) in
  let net_n = net_n(ttl1, seq1, addr_d, iv) in
  let nkencd = AES_CCM((t2, lpct), enck, net_n) in
  let pecb = e(pri_key, concat(iv, nkencd)) in
  out(ch, (obfuscate(net_n, pecb), nkencd));
  //Receive friend offer
  in(ch, (obfs: bitstring, d: bitstring));
  let pecb = e(pri_key, concat(iv, d)) in
  let network_n = deobfuscate(obfs, pecb) in
  let seq2 = secondconcat9(network_n) in
  let friaddr = thirdconcat9(network_n) in
  let (=t1, frict: bitstring) = sdec(d, enck,
    network_n) in
  let t = concat(concat(concat(addr_d, friaddr),
    lpct), frict) in
  insert mesh_friend_key_lp(friaddr,
    enck(k2(netkey, t)), prik(k2(netkey, t)))).
```

TABLE 5: Result, corresponding attack (if violated), and runtime of the property verification. ✓: property holds; ✗: the violation is caused by known vulnerability.

Property	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**Fix**
P1	✗/I/13s	✗/I/6s	✗/I/3s	✗/I/3s	✗/I/20s	✗/I/12s	✗/I/4s	✗/M/8s	✓/–/8s
P2	✓/–/14s	✓/–/6s	✓/–/3s	✓/–/3s	✓/–/29s	✓/–/13s	✓/–/8s	✗/M/9s	✓/–/8s
P3	✓/–/12s	✓/–/5s	✓/–/3s	✓/–/2s	✗/I/23s	✗/I/10s	✗/I/4s	✗/M/9s	✓/–/8s
P4	✓/–/12s	✓/–/5s	✓/–/3s	✓/–/2s	✓/–/20s	✓/–/9s	✓/–/4s	✗/M/11s	✓/–/8s
P5	✓/–/1h33m58s	✓/–/53m1s	✓/–/13m51s	✓/–/13m19s	✗/I/279h32m11s	✗/I/44h4m53s	✗/I/10h25m27s	✗/M/307h2m15s	✓/–/18m11s
P6	✓/–/13s	✓/–/7s	✓/–/3s	✓/–/4s	✗/I/54s	✗/I/27s	✗/I/10s	✗/M/7s	✓/–/9s
P7	✓/–/14s	✓/–/7s	✓/–/3s	✓/–/3s	✗/I/58s	✗/I/24s	✗/I/8s	✗/M/8s	✓/–/9s
P8	✓/–/1h32m4s	✓/–/50m56s	✓/–/13m46s	✓/–/14m	✗/I/292h11m23s	✗/I/42h8m53s	✗/I/10h16m16s	✗/M/95h12m42s	✓/–/19m14s
P9	✓/–/1h32m32s	✓/–/50m26s	✓/–/13m52s	✓/–/14m19s	✗/I/282h6m4s	✗/I/42h27m7s	✗/I/12h47m3s	✗/M/96h39m12s	✓/–/18m56s
P10	✓/–/27s	✓/–/20s	✓/–/7s	✓/–/12s	✗/I/3m43s	✗/I/1m48s	✗/I/38s	✗/M/2m26s	✓/–/20s
P11	✓/–/27s	✓/–/20s	✓/–/7s	✓/–/11s	✗/I/3m40s	✗/I/1m45s	✗/I/35s	✗/M/2m33s	✓/–/20s

I: The impersonation attack presented in the existing work (BlueMAN [13] and BlueMirror [21]). M: MitM attack.