BLURtooth: Exploiting Cross-Transport Key Derivation in Bluetooth Classic and Bluetooth Low Energy

Anonymous Submission #129

Abstract

The Bluetooth standard specifies two incompatible wireless transports: Bluetooth Classic (BT) for high-throughput services and Bluetooth Low Energy (BLE) for very low-power services. BT and BLE have different security architectures and threat models, but they use similar security mechanisms. In particular, pairing enables two devices to establish a long term key to secure the communication. Two devices have to pair over BT and BLE to use both transports securely. Since pairing the same devices two times is considered "user-unfriendly", Bluetooth v4.2 introduced Cross-Transport Key Derivation (CTKD). CTKD allows two devices to pair once, either over BT or BLE, and generate both BT and BLE long term keys. Despite CTKD allowing to cross the security boundary between BT and BLE, the security implications of CTKD have not yet been investigated.

We present the first security analysis of CTKD and identify five cross-transport issues for BT and BLE. These issues enable, for the first time, exploitation of both BT and BLE by attacking either transport. Based on the identified issues, we demonstrate four novel cross-transport attacks resulting in device impersonation, traffic manipulation, and malicious session establishment. We refer to them as BLUR attacks, as they blur the security boundary between BT and BLE. The BLUR attacks are standard-compliant and therefore apply to all devices supporting CTKD, regardless of implementation details. We successfully demonstrate the BLUR attacks on 13 devices with 10 unique Bluetooth chips, and discuss effective countermeasures. We disclosed our findings and countermeasures to the Bluetooth SIG in May 2020.

1 Introduction

Bluetooth is a pervasive wireless technology used by billions of devices including mobile phones, laptops, headphones, cars, speakers, medical, and industrial appliances [13]. Bluetooth is specified in an open standard maintained by the Bluetooth special interest group (SIG), and the latest version of the standard is 5.2 [12]. The standard specifies two incompatible wireless transports, Bluetooth Classic (BT) and Bluetooth Low Energy (BLE). BT is best suited for high-throughput use cases, such as streaming audio and voice calls, while BLE is best suited for very low-power use cases such as localization and monitoring.

As BT and BLE were introduced at different point in time to address different use cases, the standard maintains *separate security architectures and threat models* for BT [12, p. 947] and BLE [12, p. 1617]. While these security architectures address different threat models, they use similar security mechanisms, including pairing and secure session establishment. Pairing enables devices to establish a shared long term key, and secure session establishment enables paired devices to establish a secure communication channel by negotiating a session key that is derived from the pairing long term key.

Devices that support both BT and BLE have to pair twice to use both transports securely. As pairing the same devices twice is considered "user-unfriendly", Bluetooth v4.2 (released in 2014) introduced *Cross-Transport Key Derivation (CTKD)*. After pairing on one transport, CTKD allows the creation of a second long term key for the other transport [12, p. 1401]. For example, two devices can pair over BT, generate the BT long term key, and then run CTKD to derive the BLE long term key (without having to pair over BLE). All major Bluetooth software stacks (Apple, Linux, Android, and Windows) and hardware providers (Cypress, Intel, Qualcomm, Broadcom, Apple, Sony, and Bose) implement CTKD. Apple presented CTKD as a core "always on" Bluetooth feature to improve usability [42].

CTKD is a promising attack target as it crosses the security boundary between BT and BLE (i.e., when using CTKD, pairing over one transport automatically provides security guarantees for both transports). Despite this fact, the security of CTKD remains unexplored. For example, the standard does not include CTKD in the BT and BLE threat models and we are not aware of any security analyses of CTKD. So far, all existing attacks focused exclusively on either transport.

We present the first security analysis of CTKD, uncovering five security issues. Our issues are novel because they are the first examples of cross-transport issues for Bluetooth. Based on those issues, we demonstrate four cross-transport attacks, enabling impersonation, interception, and manipulation of traffic between victims, as well as unintended device sessions. Our attacks are standard-compliant and are thus effective against all devices that support CTKD. As we are the first to exploit CTKD and enable BT and BLE cross-transport exploitation, the attacks are orthogonal to other standard-compliant attacks on BT and BLE [1,3,4,10,21,22,35,38]. We name our attacks *BLUR* attacks, as they blur the security boundary of BT and BLE.

We implement the BLUR attacks using a widely available Bluetooth development board connected to a laptop running Linux and developing custom software based on open-source tools. This makes reproducing the BLUR attacks simple and affordable. Our evaluation demonstrates that all tested devices are vulnerable. We will release our tools to the public after the responsible disclosure process completes. We use our attack implementation to evaluate 13 devices, with 10 unique Bluetooth chips, from the major hardware and software vendors, e.g., Broadcom, Cambridge Silicon Radio (CSR), Cypress, Google, Intel, Linux, Qualcomm, and Windows and representing all Bluetooth versions that support CTKD (i.e., 4.2, 5.0, and 5.1) and even a device supporting Bluetooth version 4.1 to which CTKD has been backported.

We summarize our contributions as follows:

- We perform the first security analysis of CTKD (Section 3), and show that it enables to cross the security boundary between BT and BLE. We identify five novel and very serious issues, which enable the first cross-transport attacks between BT and BLE.
- We propose four attacks to exploit the issues in CTKD (Section 4). Our attacks allow impersonation, interception, traffic manipulation, and unintended sessions. We present a low-cost implementation of the attacks based on a Linux laptop and a Bluetooth development board.
- We confirm that real-world BT and BLE devices are vulnerable to the BLUR attacks by evaluating our attacks on 13 unique devices (Section 5). We provide mitigation strategies to address the attacks directly in the Bluetooth standard. We have disclosed our findings and mitigations to the Bluetooth SIG in May 2020.

2 Background

We now compare BT and BLE, and introduce CTKD.

2.1 A Comparison of BT and BLE

BT and BLE are two wireless transports specified in the Bluetooth standard. These transports are incompatible (i.e., while they use the same 2.4 GHz band the physical and link layers are different) and are designed to complement each other. BT is used for high-throughput and connection-oriented services, such as streaming audio and voice. BLE is used for very low-power and low-throughput services such as localization and monitoring. Typically, high-end devices, such as laptop, smartphones and tablets, provide BT and BLE (in a single radio chip), while low end devices such as mice, keyboards and wearables provide either BT or BLE.

BT and BLE have similar security mechanisms but different security architectures and threat models. Both transports provide a pairing mechanism, named Secure Simple Pairing (SSP), to let two devices establish a long term key. During pairing, BLE allows negotiating the entropy of the long term key while BT does not. Both transports provide a secure session establishment mechanism to derive a session key from the long term key and protect the communication. During session establishment, BT allows negotiating the entropy of the session key while BLE inherits the entropy of the session key from the entropy of the long term key.

BT and BLE support a “Secure Connections” mode that uses FIPS compliant security primitives such as AES-CCM for authenticated encryption, ECDH on P-256 for key agreement, mutual authentication procedures for the long term key, and AES-CMAC for keyed hashing. BT and BLE have similar association mechanisms that can be used to protect the pairing phase against man-in-the-middle attacks. Two examples of associations are “Just Works” that provides no protection and “Numeric Comparison” that provides protection against man-in-the-middle attacks by requiring user interaction (e.g., the user has to manually confirm that she sees the same numeric code on the pairing devices).

BT and BLE define master and slave roles in different ways. For BT, the master is the connection initiator, the slave is the connection responder, and roles can be switched. Both master or slave can request a role switch almost anytime after a radio link between the two is established. For BLE, master and slave roles are fixed and switching roles is not supported. The master acts as the connection initiator (BLE central) and the slave as the connection responder (BLE peripheral). High-end BLE devices, such as laptops and smartphones, implement both master and slave modes and are typically used as the master, while low-end devices, such as fitness trackers or smartwatches, implement only the slave mode.

2.2 Cross-Transport Key Derivation (CTKD)

Two devices that support BT and BLE have to pair over BT and over BLE to use both transports securely. Pairing the same two devices two times is not user-friendly and the Bluetooth standard addressed this issue in Bluetooth 4.2 (released in 2014) by introducing CTKD. As shown in Figure 1, CTKD enables two devices to pair once, either over BT or BLE, and then securely use both [12, p. 280]. For example, a user can

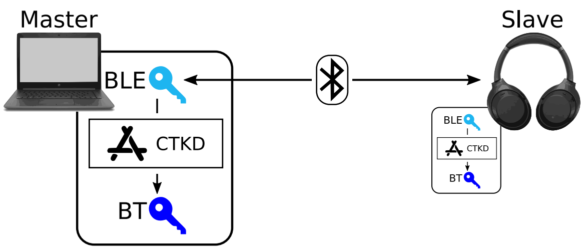

Figure 1: CTKD overview. CTKD is used by two devices who paired and share a long term key over BLE to derive a long term key for BT. CTKD can also be used to derive BLE pairing keys after two devices paired over BT.

pair a headset and a laptop over BLE, without putting the headset in BT discoverable mode, and then securely connect the headset and the laptop over BT (without having to pair over BT). It is also possible to do the initial pairing over BT, and use CTKD to generate the BLE pairing key.

Before explaining how does CTKD work, is important to review the differences between *pairable (bondable)* and *discoverable* states for BT and BLE. If a device is pairable then is going to accept to pair with other devices, while if it is discoverable is going to reveal his identity to other devices. It is widely believed that a device is required to be discoverable and pairable to be able to pair, however only the pairable state is required. For example, when pairing a laptop with a pair of headphones over BT, typically only the headphones are discoverable and pairable and the laptop is only pairable. Hence, it is possible to pair with a device even if it is not discoverable [41].

The Bluetooth standard specifies the same CTKD function to derive BT and BLE long term keys. This function takes as inputs a 16-byte key and two 4-byte strings and derives a 16-byte key using AES-CMAC (see Section 5.4 for CTKD's internals). What changes between BT and BLE are the inputs to the CTKD function. BT derives a BLE long term key (K_{BLE}) from a BT long term key (K_{BT}) and the strings "`tmp2`" and "`brle`". While BLE derives K_{BT} from K_{BLE} and the strings "`tmp1`" and "`lebr`". As the standard defines constant strings and no fresh nonces as inputs, the CTKD function derives the same output key when reusing the same input key.

CTKD is widely supported by vendors such as Apple [42], Google [5], Cypress [15], Linux [14], Qualcomm [33], and Intel [23]. CTKD is used in combination with "Secure Connections", that is a security mode that was introduced to enhance the security primitives of BT and BLE without affecting their security mechanisms. For example, "Secure Connections" introduced AES-CCM authenticated-encryption for BT, and ECDH pairing for BLE.

3 Security Analysis of CTKD

BT and BLE are incompatible wireless technologies with different security architectures and threat models (see Section 2.1). CTKD, as shown in Figure 1, improves BT and BLE usability by allowing two devices to pair once, either over BT or BLE, and compute the keying material for both transports without requiring a second pairing (see Section 4).

CTKD is a critical attack target for several reasons. Firstly, CTKD crosses the security boundary between BT and BLE. Hence, if CTKD introduces a vulnerability on one transport, that vulnerability is exploitable for both BT and BLE. Secondly, CTKD is applicable to "Secure Connections", the most secure mode for BT and BLE. Hence, if CTKD is vulnerable then the attacker can break Bluetooth "Secure Connections".

Despite the potential security risks related to CTKD, the Bluetooth standard does not provide a security analysis of CTKD and does not include CTKD in the BT and BLE threat models [12, p. 1401]. We address this concern by performing the first security analysis of CTKD and uncovering five cross-transport issues. A cross-transport issue enables an attacker to exploit BT from BLE or BLE from BT. This category of issues is novel in the context of Bluetooth. We now introduce our system and attacker models, and then we describe the identified cross-transport issues using a reference example.

3.1 System Model

Our system model considers two victims, Alice and Bob, who want to securely communicate over BT and BLE. Alice and Bob support CTKD and during pairing and session establishment propose the strongest security mechanisms (e.g., SSP, "Secure Connections", and "Numeric Comparison"). Such mechanisms are expected to protect Alice and Bob against impersonation, eavesdropping, and man-in-the-middle attacks on BT and BLE. Alice and Bob can run secure sessions both over BT and BLE. Without loss of generality, we assume that Alice is the BT and BLE master and Bob is the BT and BLE slave.

Regarding the notation, we indicate a BT long term key with K_{BT}, a BT session key with SK_{BT}, a BLE long term key with K_{BLE}, and a BLE session key with SK_{BLE}. We indicate a Bluetooth address with ADD. Furthermore, we indicate a public key with PK, a private key with SK, a nonce with N, and a message authentication code with MAC.

3.2 Attacker Model and Goals

Our attacker model considers Charlie, a remote attacker in Bluetooth range with Alice and Bob. The attacker aims to compromise the secure BT and BLE sessions between the victims. The attacker's knowledge is limited to what Alice and Bob advertise over the air, e.g., full or partial Bluetooth addresses, Bluetooth names, authentication requirements, IO capabilities, and device classes. The attacker does not know long term keys or session keys shared between Alice and Bob and does not observe Alice and Bob when they pair or establish a secure session. Regarding the attacker's capabilities, the attacker can scan and discover BT and BLE devices, jam the Bluetooth spectrum, pair over BT and BLE using CTKD, propose weak association mechanisms (e.g., "Just Works"), and dissect and craft unencrypted Bluetooth packets.

The attacker has four goals. The first goal is to impersonate Alice (to Bob) and take over Alice's secure sessions. The second goal is to impersonate Bob (to Alice) and take over Bob's secure sessions. Master and slave impersonations are two different goals as they require different attacks. The third goal is to establish a man-in-the-middle position in a secure session between Alice and Bob. The third goal requires combining and synchronizing the impersonation attacks on Alice and Bob. The fourth goal is to pair and establish unintended sessions with Alice or Bob as an arbitrary device, without breaking their secure session.

3.3 Cross-Transport Issues with CTKD

We now present five *cross-transport issues (CTI)* that we identified as a result of our security analysis of CTKD. The order in which we present the issues follows the life-cycle of a Bluetooth connection, with discovery, pairing, and communication phases (see Figure 2). Section 4 then presents how to leverage the issues for attacks.

CTI 1: Roles During the discovery phase, Alice and Bob can discover each other both over BT and BLE. This is a consequence of CTKD as it enables more ways to pair devices with less user interaction. Alice, as master, is expected to send pairing requests over BT or BLE to Bob, and the user expects to pair Alice and Bob by discovering Bob on Alice's screen and sending a pairing request to Bob. However, BT master and slave roles are not fixed (unlike BLE) and Alice can receive pairing requests over BT. The attacker can take advantage of this role asymmetry to impersonate a slave device that is already trusted by Alice and send *a pairing request* to Alice over BT even if Alice is expecting to receive only BT and BLE *pairing responses*.

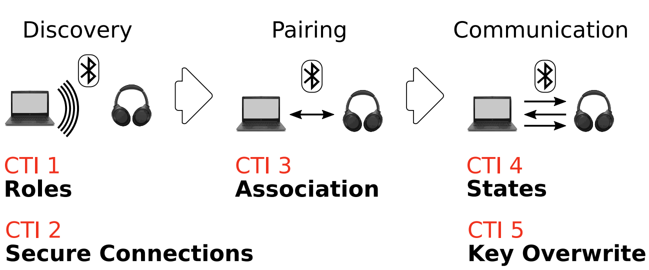

Figure 2: Cross-transport issues (CTI) with CTKD. We identify each issue with a particular phase of the Bluetooth secure connection life-cycle, but they are all directly or indirectly caused by the CTKD mechanism.

CTI 2: Secure Connections When Alice and Bob have discovered each other, they exchange their capabilities before starting the pairing process. To use CTKD they declare "Secure Connections" support for the transport used for pairing (BT or BLE). However, the specification does not specify if CTKD support requires "Secure Connections" support *only for the pairing transport* or *for both transports*. From our experiments we find that CTKD is used when "Secure Connections" is only supported by the pairing transport. This issue considerably increases the CTKD attack surface, as an attacker is not limited to target only devices which support BLE and BT "Secure Connections".

CTI 3: Association After exchanging their capabilities, Alice and Bob perform the pairing process. BT and BLE pairings are fundamentally different but they provide similar association mechanisms. However, the choice of the association mechanism is not enforced across BT and BLE. This issue can be exploited by the attacker to pair with a weak association mechanism, such as "Just Works", on one transport while the other transport expects a strong association mechanism, such as "Numeric Comparison". This is especially dangerous in case of impersonation attacks because the user is not going to notice an attacker that is (re-)pairing using "Just Works" pretending to be a previously securely paired and trusted device.

CTI 4: States When Alice and Bob complete pairing they *remain pairable over BT and BLE* and Bob even *remains discoverable* over BLE. This is not the case without CTKD where a device is pairable and optionally discoverable only on one transport. This issue gives the attacker more options to discover and pair with victim devices. For example, the attacker can pair on the transport that is not currently in use by Alice and Bob. Furthermore, in some CTKD use cases one transport is supposed to be used only for pairing and deriving keys for the other. Hence, that transport is always in pairable state but never used after paring. This enables the attacker to establish unintended malicious sessions on both transports by pairing on the unused one and forcing CTKD.

CTI 5: Key Overwrite Once Alice and Bob are paired, they share a long term key, derive a second long term key via CTKD, and start a secure channel on BT and/or BLE. If Alice and Bob already shared a long term key for the transport used by CTKD then *CTKD will overwrite the existing key*. This is a serious issue because an attacker who is impersonating either Alice or Bob can use CTKD to overwrite long term keys. For example, if Alice and Bob are running a secure session over BT then the attacker can pair with Bob over BLE while impersonating Alice and overwrite the BT key that is shared by Alice and Bob.

4 BLUR Attacks on CTKD

We now design four novel CTKD cross-transport attacks based on the five cross-transport issues that we discuss in Section 3.3. Our attacks are the first attacks that exploit CTKD by blurring the security boundary between BT and BLE. These attacks are standard-compliant and enable impersonation, interception, and manipulation of traffic between victims, as well as unintended sessions with a victim device. We call our attacks *BLUR attacks*.

4.1 BLUR Impersonation Attacks

Figure 3 presents the BLUR impersonation attack strategy using a slave impersonation attack as a reference example. Before the attack takes place Alice and Bob (the victims) are running a secure BT session and they share a BT long term key (K_{BT}). Charlie (the attacker), targets the BLE transport (which is not used by the victims) and pairs with Bob over BLE, pretending to be Alice. Because of CTKD, Bob overwrites the BT long term key that he established with Alice with the one derived when pairing with Charlie. As a result, Charlie takes over Alice's BT session, and Alice can no longer connect to Bob as she does not possess the correct K_{BT}.

In the following two paragraphs we describe the technical details of the BLUR master and slave impersonation attacks.

Master impersonation Charlie impersonates Alice (master) and takes over her BT secure session with Bob as in Figure 4. Charlie sends a BLE pairing request using Alice's Bluetooth address (ADD_A) and requests to use "Just Works" to avoid user interaction. The BLE pairing request is standard-compliant because Charlie impersonates a BLE master and is going to be accepted by Bob who is pairable over BLE. Bob sends a BLE pairing response believing that Alice wants to

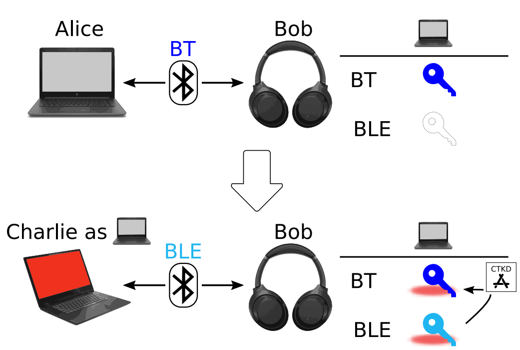

Figure 3: BLUR impersonation attack strategy. Charlie pairs with Bob over BLE, overwriting Alice's key BT.

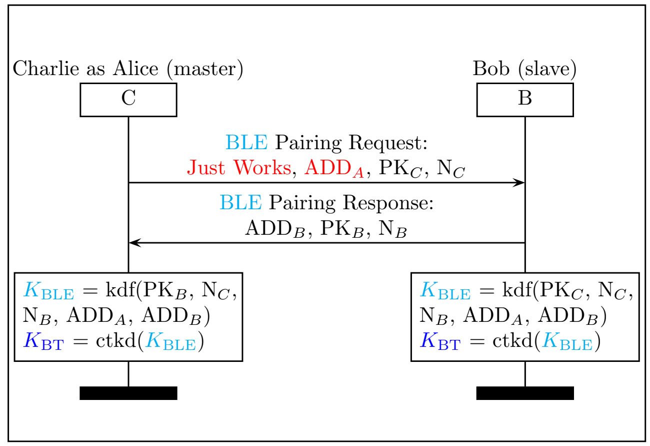

Figure 4: Master impersonation attack and takeover. Charlie (acting as master) pairs with Bob over BLE, overwriting Alice's key.

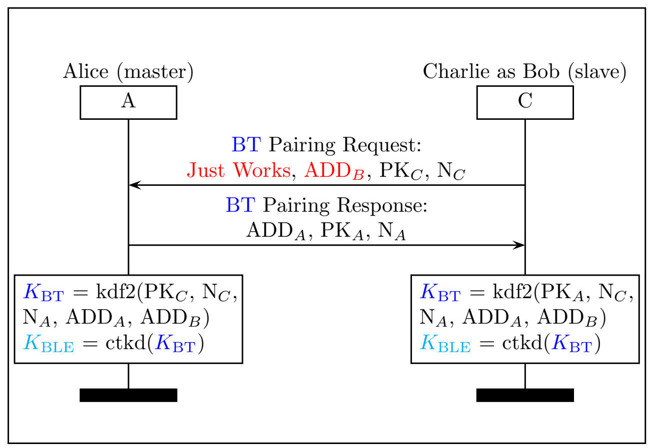

Figure 5: Slave impersonation attack and takeover. Charlie (acting as slave) sends a BT pairing request to Alice (master) as Bob, overwriting Bob's key.

pair (or repair) over BLE. Charlie and Bob use the exchanged nonces and public keys to compute K_{BLE} (kdf) and derive K_{BT} from K_{BLE} using CTKD's key derivation functions (ctkd). As a result of the master impersonation attack, Charlie forces Bob to overwrite the BT pairing key that he established with Alice with his BT key, shares a BLE key with Bob, and takes over Alice's BT session. Alice can no longer establish secure sessions with Bob as, during pairing with Charlie, Bob overwrote her shared ley.

Slave impersonation Charlie impersonates Bob (slave) and takes over his BT secure session with Alice as in Figure 5. Charlie sends a BT pairing request using Bob's Bluetooth address (ADD_B) and requests to use "Just Works" to avoid user interaction. The BT pairing request is standard-compliant because BT allows a slave to send a BT pairing request by

	CTI 1 Roles	CTI 2 SC	CTI 3 Association	CTI 4 States	CTI 5 KO
Master Imp.	x	✓	✓	✓	✓
Slave Imp.	✓	✓	✓	x	✓
MitM	✓	✓	✓	✓	✓
Unin. Sess.	x	✓	x	✓	x

Table 1: The mapping between the five cross-transport issues (CTI) identified in Section 3.3 and the four BLUR attacks discussed in Section 4. We abbreviate "Secure Connections" with SC, and Key Overwrite with KO.

switching role before pairing is started. Alice, who is pairable over BT, sends a BT pairing response believing that she is talking to Bob (a trusted device). Charlie and Alice use the exchanged nonces and public keys to compute K_{BT} (kdf2), and derive K_{BLE} from K_{BT} using CTKD's key derivation functions (ctkd). As a result of the slave impersonation attack, Charlie forces Alice to overwrite the BT pairing key that she established with Bob with his BT key, shares a BLE key with Alice, and takes over Bob's BT session. Bob cannot re-establish secure sessions with Alice as he no longer possess the correct paring keys.

As summarized in Table 1, the master impersonation attack takes advantage of all the cross-transport issues that we present in Section 3.3 except CTI 1. In particular, the attacker takes advantage of non-consistent "Secure Connections" support (CTI 2), lack of consistency between BT and BLE association methods (CTI 3), more opportunities to pair (CTI 4), and key overwriting (CTI 5). The slave impersonation attack takes advantage of all CTIs except CTI 4, including the role asymmetries between BT and BLE (CTI 1).

4.2 BLUR Man-in-the-Middle Attack

Figure 6 presents the high-level description of our BLUR man-in-the-middle attack. As in the previous section, Alice and Bob are paired over BT and they run a secure session over BT. During this attack, Charlie sequentially performs the master

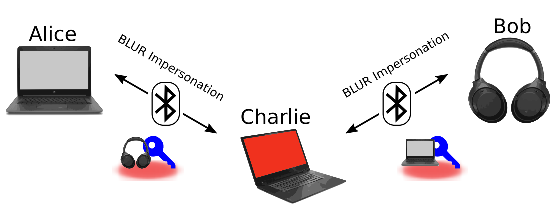

Figure 6: BLUR man-in-the-middle attack. The attacker uses the BLUR Impersonation attack against two devices that were previously paired. The two devices do not detect a change but Charlie now has access to all traffic.

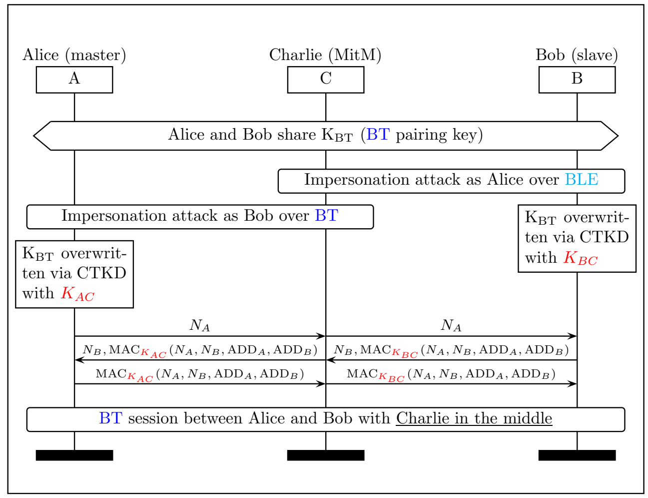

Figure 7: MitM attack and takeover. Charlie impersonates Alice as in Figure 4, impersonates Bob as in Figure 5, let the victims mutually authenticate and then gets access to their traffic.

and slave impersonation attacks described in Section 4.1. As a result, the attacker overwrites Alice and Bob's BT pairing keys with known keys, establishes BLE long term keys with Alice and Bob, and positions himself in the middle to access all traffic between the victims and to inject valid traffic both on BT and BLE.

Figure 7 shows the details of the MitM attack. Firstly, Charlie impersonates Alice to Bob over BLE (as in Figure 4), overwrites Bob's BT key with his key (K_{BC}). Secondly, Charlie impersonates Bob to Alice over BT as in Figure 5 and overwrites Alice's BT key with his key (K_{AC}). Then, Alice and Bob exchange two nonces (N_A, N_B) to authenticate the BT pairing key. Charlie mutually authenticates with Bob and Alice by using a message authentication code (MAC) function keyed with the appropriate key and input parameters. Finally, Alice and Bob establish a secure BT session with Charlie in the middle, and Charlie gets access to all traffic exchanged by Alice and Bob and can modify and inject arbitrary valid traffic between Alice and Bob.

As summarized in Table 1, the BLUR man-in-the-middle attack is a composition of the master and slave impersonation BLUR attacks and takes advantage of all the CTI that we present in Section 3.3.

4.3 BLUR Unintended Sessions Attack

Figure 8 presents a BLUR unintended session attack targeting Bob. In this scenario, Alice and Bob are running a secure session over BT but they are still pairable over BLE in order to accept pairing requests with other devices and run CTKD. Charlie targets Bob (slave) by sending him a paring request over BLE as an unknown device. Charlie can pretend to be any

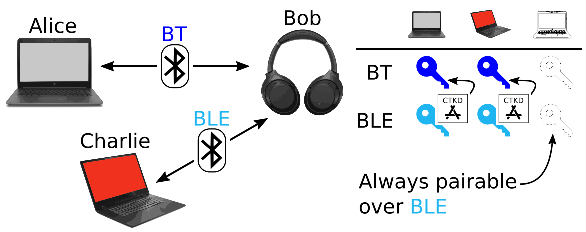

Figure 8: BLUR unintended sessions attack. Charlie sends a BLE pairing request to Bob (who remains pairable over BLE due to CTKD) as an unknown device with arbitrary capabilities. After CTKD completes, Charlie can establish secure but unintended BT and BLE sessions with Bob without breaking Bob's existing pairings and sessions.

device having arbitrary capabilities, e.g., Bluetooth address, Bluetooth name, device class, "Secure Connections" support, and weak association. Bob, accepts to pair with Charlie while continuing his session with Alice. Then, Charlie and Bob negotiate K_{BLE}, and derive K_{BT} using CTKD. Now, Charlie can establish secure but unintended BT and BLE sessions with Bob without breaking his existing pairings or sessions with other devices (e.g., with Alice).

Charlie can also establish unintended sessions with Alice (master). In particular, he can impersonate a BLE slave and start advertising his presence. Once Alice discovers Charlie, she can establish a BLE connection with him, and Charlie can explicitly request to pair using a SMP Security Request packet [12, p. 1401]. Then, Alice and Charlie compute K_{BLE}, and derive K_{BT} using CTKD. Now, Charlie can establish secure but unintended BT and BLE sessions with Alice without breaking her existing pairings or sessions with other devices (e.g., with Bob). Charlie can take advantage of the unintended sessions with Alice and Bob in many ways. For example, he can use the session to drop known exploits such as Blue-Borne [6], BLEEDINGBIT [7], or SweynTooth [20], new exploits, and to enumerate and tamper with BT and BLE services and characteristics (including the protected ones).

Those attacks are particularly effective when the victims are using one transport only to pair and derive keys with CTKD. For example, a Bluetooth speaker only streams music over BT but is also pairable over BLE to enable users to discover it without having to put it into BT pairing mode. As summarized in Table 1 the unintended session BLUR attack takes advantage of CTI 2 and CTI 4.

5 Implementation and Evaluation

In this section we describe our attack scenario, our attack device, an implementation of the BLUR attacks (proposed in Section 4) using open-source software and off-the-shelf hardware, an implementation of the CTKD mechanism used to validate our attacks, and an evaluation of our attacks on 13 unique devices from different hardware and software vendors. The tools that we developed will be open-sourced after responsible disclosure with the Bluetooth SIG.

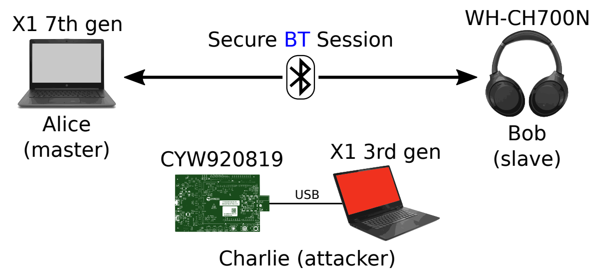

Figure 9: BLUR Attack Scenario. Alice (master) is a ThinkPad X1 7th gen, Bob (slave) is a pair of Sony WH-CH700N headphones and Charlie (attacker) is a CYW920819 board connected via USB to a ThinkPad X1 3rd gen. Alice and Bob have paired in absence of Charlie, and are running a secure BT session.

5.1 BLUR Attack Scenario

Our attack scenario is presented in Figure 9 and includes two victims, a 7th generation ThinkPad X1 laptop (Alice, master) and a pair of Sony WH-CH700N headphones (Bob, master). The attacker (Charlie) uses a CYW920819 development board [16] and a 3rd generation ThinkPad X1 laptop. As in our attack descriptions in Section 4, the victims have securely paired in absence of Charlie, and are running a secure BT session. The evaluation results presented in Section 5.6 are obtained by using the same attack scenario targeting different victim devices.

Table 2 presents the relevant Bluetooth features supported by Alice and Bob. We note that Bob is capable of using CTKD over BLE even if he does not support "Secure Connections" over BT and does not support Bluetooth version 4.2. This confirms the "Secure Connections" cross-transport issue (CTI 2) that we discuss in Section 3.3. Table 2 also shows the features supported by Charlie, and indicates with an asterisk (*) the features that we can modify with our implementation of the BLUR attacks. For example, our implementation enables to send pairing requests over BT and BLE with with arbitrary Bluetooth addresses, names, associations, "Secure Connections" (SC) support, and authentication requirements (AuthReq).

5.2 BLUR Attack Device

Our attack device consists of a Linux laptop (Bluetooth host) connected to a CYW920819 development board (Bluetooth controller). We implement our attack device by developing custom code and tools both for Linux and the board. Regarding the host, we modify and recompile the Linux kernel and

	Alice	Bob	Charlie
Device(s)	X1 7th gen	WH-CH700N	X1 3rd gen / CYW920819
Radio Chip	Intel	CSR	Intel / Cypress
Subversion	256	12942	256 / 8716*
Version	5.1	4.1	5.0*
Name	x7	WH-CH700N	x1*
ADD	Redacted	Redacted	Redacted*
Class	0x1c010c	0x0	0x0*
BT SC	True	Only Controller	True*
BT AuthReq	0x03	0x02	0x03*
BLE SC	True	True	True*
BLE AuthReq	0x2d	0x09	0x2d*
CTKD	True	True	True*
h7	True	False	True*
Role	Master	Slave	Master*
IO	Display	No IO	Display*
Association	"Numeric C."	"Just Works"	"Numeric C."*
Pairable	True	True	True*

Table 2: Relevant Bluetooth features for Alice, Bob, and Charlie. Alice and Bob support CTKD even if Bob's Host does not support BT SC (BT "Secure Connections"). We redact the devices' Bluetooth addresses for privacy reasons. We append an asterisk (*) to the attacker's features that we can modify with our implementation.

BlueZ according to our needs. For example, by changing the kernel we enable parsing of diagnostic messages from the controller, and by changing BlueZ we can develop custom user-space management commands for BT and BLE.

Regarding the controller, we use the board's proprietary patching mechanism to modify the Bluetooth firmware according to our needs. For example, by writing the firmware's RAM we can change the attack device's features, including the features containing an asterisk (*) in Table 2. This process required significant engineering effort as we had to dump the Bluetooth firmware from the board, reverse-engineer the relevant functions and data structures, and write and test our ARM assembly patches.

Our attack device makes use of several free and open-source tools to automate the configuration and management of BT, BLE, and the BLUR attacks. Table 3 presents the list of such tools with a brief description of their usage. Overall, our usage of low-cost hardware and open-source software will enable other researchers to easily reproduce the BLUR attacks.

Tool	Usage
`ghidra`	RE the devboard firmware [39]
`internalblue`	Patch devboard firmware [30]
`wireshark`	Monitor HCI, LMP, and SMP
`hciconfig`	Configure HCI interfaces
`hcitool`	Scan, connect and enumerate BLE devices
`bleah`	Scan, connect and enumerate BLE devices
`scapy`	Craft and decode packets [11]
`pybt`	Custom BLE pairing [36]
`linux414`	Modify BLE pairing capabilities
`bluez`	Modify Linux userspace configuration
`pybluez`	Test BT and BLE using the BlueZ API
`scapy`	Configure HCI, manage BT and BLE sockets
`bluetoothctl`	Manage, pair and connect devices
`btmgmt`	Manage, pair and connect devices

Table 3: Open-source tools used to implement the BLUR attacks.

5.3 BLUR Attacks Implementation

The BLUR attacks, presented in Section 4, include master impersonation, slave impersonation, man-in-the-middle, and unintentional session attacks. In the next paragraphs, we describe how we implemented them based on our attack device in the attack scenario presented in Section 5.1.

Laptop (master) impersonation attack To impersonate the laptop, we configure our attack device to clone the laptop Bluetooth features, including Bluetooth address, Bluetooth name, device class, BT and BLE "Secure Connections" support, and advertised services. We accomplish this task by patching the attack device's Bluetooth firmware and configuring the attack laptop accordingly. Once the attack device looks like the impersonated laptop, we ask the headphones to pair over BLE using "Just Works" and CTKD.

The malicious BLE pairing request is sent using `btmgmt`'s text-based user interface (TUI). The headphones accept to pair over BLE, update the BLE long term key, run CTKD for BT, update the BT long term key, and establish a secure BLE session with the attack device. Then, the headphones terminate the BT session with the impersonated laptop and establish a secure BT session with the attack device. The impersonated laptop cannot connect back with the headphones as it does not possess the new BT and BLE long term keys.

Headphones (slave) impersonation attack To impersonate the headphones, we configure our attack device to clone the headphones Bluetooth features using the same technique adopted for the laptop impersonation. Once the attack device looks like the impersonated headphones we ask the laptop to pair over BT using "Just Works" and CTKD. The malicious BT pairing request is sent using `btmgmt`'s TUI. The laptop accepts to pair over BT, updates the BT long term key, and runs CTKD for BLE. Then, we establish a secure BT session with the headphones.

Man-in-the-middle attack By using our BLUR implementation with two development boards connected to the same attack laptop, we can impersonate the laptop and the headphones at the same time, and man-in-the-middle them. In particular, we run the laptop (master) impersonation attack first, and then the headphone (slave) impersonation attack. As a result, the attack device positions itself in the middle between the victims.

Unintended sessions attack To perform the unintended sessions attacks, we configure the attack device to impersonate an arbitrary device with arbitrary services over BT and BLE. Then we send a malicious pairing request to the headphones over BLE and one to the laptop over BT. Both pairing requests declare support for CTKD and "Just Works". The attack device establishes new BT and BLE keys both with the headphones and the laptop and starts unintended sessions with both over BT and BLE.

5.4 CTKD Mechanism Implementation

The Bluetooth standard does not provide a reference implementation for the key derivation function used by CTKD, and provides limited documentation about its design [12, p. 1401]. We decided to implement it in Python 3 using the PyCA cryptographic module [8] and we successfully tested our implementation against the test vectors in the standard. We used our implementation to validate the BT and BLE keys derived using CTKD while performing our attacks and the code will be open-sourced. We now describe the CTKD key derivation function implementation details.

As explained in Section 2.2, the Bluetooth standard specifies a single CTKD function that is used with different parameters for BT and BLE. Figure 10 shows the CTKD key derivation function for BT (top) and BLE (bottom). Both use a chain of two AES-CMAC blocks in sequence with different keys and 4-byte constant strings. AES-CMAC is a message authentication code (MAC) based on the AES block cipher [18]. In particular, BT uses K_{BT}, "`tmp2`" and "`brle`" and derives K_{BLE}, while BLE uses K_{BLE}, "`tmp1`" and "`lebr`" and derives K_{BT}.

In the first AES-CMAC, if both devices support the h7 algorithm, the long term key is used as key and the string as input, otherwise, the string (padded with 12 zeros) is used as key and the long term key as input. In the second AES-CMAC, the 16-byte output of the first AES-CMAC is used as key and the string as input. The 16-byte output of the second AES-CMAC is the derived long term key.

5.5 BLUR Attacks Evaluation Setup

With our attack implementation (Section 5.3), we are capable of conducting all four BLUR attacks. We used the attack device both as the attacker and as one of the victims. For example, in a master impersonation attack we pair the attack device with the slave victim device, we disconnect them, we "forget" the victim device on the attack device and we run the master impersonation attack from the attack device. This setup is practical because it allows us to quickly test many slave victims. For the slave impersonation, we use the same procedure and quickly test many master victims.

If a victim device is vulnerable to the master or slave impersonation attack then is also vulnerable to the man-in-the-middle attack, as the latter requires a vulnerable master device and a vulnerable slave device. Regarding the unintended session attack, we test this attack by connecting the target victim to a third device and then by trying to establish unintended sessions with the victim as an arbitrary device over the transport that is not used by the legitimate connection. For example, if the victim is a pair of headphones that is connected with a laptop over BT then we run the unintended session attacker over BLE.

5.6 BLUR Attacks Evaluation Results

We evaluated the BLUR attacks on 13 devices, and Table 4 shows our evaluation results. The first six columns indicate the device producer, device model, OS, chip manufacturer, chip model, and supported Bluetooth version. The seventh column indicates the attacker role. The last three columns contain a checkmark (✓) if a device is vulnerable to the master Impersonation attack (MI), slave impersonation attack (SI), man-in-the-middle attack (MitM), or unintended session (US) attack. The master and slave impersonation attacks are grouped in one column (MI/SI column). If the victim's role

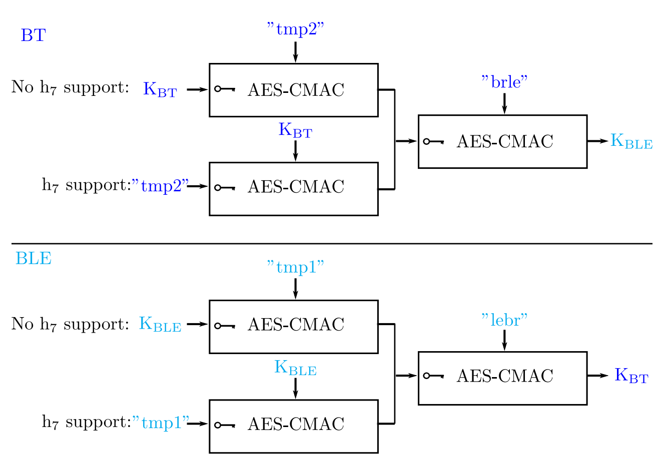

Figure 10: CTKD key derivation function for BT (top) and BLE (bottom).

Device			Chip		Bluetooth	BLUR Attack			
Producer	Model	OS	Producer	Model	Version	Role	MI/SI	MitM	US
Cypress	CYW920819EVB-02	Proprietary	Cypress	CYW20819	5.0	Slave	✓	✓	✓
Dell	Latitude 7390	Win 10 PRO	Intel	8265	4.2	Slave	✓	✓	✓
Google	Pixel 2	Android	Qualcomm	SDM835	5.0	Slave	✓	✓	✓
Lenovo	X1 (3rd gen)	Linux	Intel	7265	4.2	Slave	✓	✓	✓
Lenovo	X1 (7th gen)	Linux	Intel	9560	5.1	Slave	✓	✓	✓
Samsung	Galaxy A40	Android	Samsung	Exynos 7904	5.0	Slave	✓	✓	✓
Samsung	Galaxy A51	Android	Samsung	Exynos 9611	5.0	Slave	✓	✓	✓
Samsung	Galaxy A90	Android	Qualcomm	SDM855	5.0	Slave	✓	✓	✓
Samsung	Galaxy S10	Android	Broadcom	BCM4375	5.0	Slave	✓	✓	✓
Samsung	Galaxy S10e	Android	Broadcom	BCM4375	5.0	Slave	✓	✓	✓
Samsung	Galaxy S20	Android	Broadcom	BCM4375	5.0	Slave	✓	✓	✓
Sony	WH-1000XM3	Proprietary	CSR	12414	4.2	Master	✓	✓	✓
Sony	WH-CH700N	Proprietary	CSR	12942	4.1	Master	✓	✓	✓

Table 4: BLUR attacks evaluation results. The last three columns contain a checkmark (✓) if a device is vulnerable to the master Impersonation attack (MI), slave impersonation attack (SI), man-in-the-middle attack (MitM), or unintended session (US) attack. If the victim's role is slave then we test it against a master impersonation attack, otherwise, we test it against slave impersonation attack and we group the attacks in one column (MI/SI column). As shown by the last three columns, all the tested 13 devices (10 unique Bluetooth chips) are vulnerable to the relevant BLUR attacks.

is slave then we test it against a master impersonation attack, otherwise, we test it against a slave impersonation attack. As shown by the last three columns, all the 13 devices (10 unique Bluetooth chips) that we tested are vulnerable to the relevant BLUR attacks.

Our list of vulnerable devices is from a broad set of device producers (Samsung, Dell, Google, Lenovo, and Sony), operating system producers (Android, Windows, Linux, and proprietary OSes), and Bluetooth chip producers (Broadcom, CSR, Cypress, Intel, Qualcomm, and Samsung). Our evaluation demonstrates that the BLUR attacks are practical, standard-compliant, and affects all the Bluetooth versions that support CTKD (i.e., Bluetooth versions $\geq$ 4.2). As the BLUR attacks are standard-compliant, potentially all standard-compliant devices supporting CTKD are also vulnerable. Based on our evaluation, we suggest the Bluetooth SIG to fix the issues that we uncover in CTKD and we provide our set of countermeasures for the Bluetooth standard in Section 6.2.

6 Discussion

We now discuss the lessons learned, and our set of countermeasures to mitigate the BLUR attacks.

6.1 Lessons Learned

One key lesson that we learned while analyzing CTKD and designing, implementing and evaluating the BLUR attacks is that designers of security mechanisms must be careful when bridging technologies with different security architectures and threat models such as BT and BLE. As demonstrated in this work, such a combination can introduce cross-transport issues that can be exploited on a large scale.

Another key lesson that we learned is that isolating cross-transport issues is challenging, as such issues manifest at the security boundary between the affected transports and usually they are not part of the threat model. Separate security analyses of the affected transport are insufficient to discover cross-transport issues. Such issues require a security analysis that considers both transports and related threat models at the same time. In particular, the analysis must take into account an attacker with cross-transport capabilities, e.g., an attacker who can take advantage of weak mechanisms on one transport to exploit both.

6.2 Countermeasures

We now present our set of countermeasures for the BLUR attacks. We recommend addressing the BLUR attack at the Bluetooth standard level, as the BLUR attacks are standard-compliant. Our countermeasures can be implemented in the Bluetooth Host (implemented in the device OS) by storing extra metadata about a trusted Bluetooth device and by using available HCI commands and events.

Disable CTKD key overwrites CTKD allows to write and overwrite BT long term keys from BLE and BLE long term keys from BT. This enables an attacker to impersonate a device and take over her existing session on one transport by attacking the other. To fix this issue, a device should disallow key overwrites with CTKD when a paired device wants to re-pair. For example, re-pairing over BT should not overwrite a BLE long term key that was securely established in the past. When a device has lost a long term key for a transport (e.g., device reset), it should explicitly re-pair on that transport.

Enforce strong association mechanisms BT and BLE do not protect the negotiation of the association mechanism and CTKD allows two devices to use different association mechanisms on different transports when pairing and re-pairing. The BLUR attack exploits this fact to re-pair with a victim device using "Just Works" even if the victim supports "Numeric Comparison". To fix this issue, a device should keep track of which BT and BLE keys are established using CTKD, record the strongest association mechanism used while pairing and enforce it for subsequent (re-)pairings.

Enforce Secure Connections In our experiments, we can use CTKD with the WH-CH700N headphones even if they only support "Secure Connections" for BLE. This should not happen as CTKD should be used only when "Secure Connections" is supported on both BT and BLE. To fix this issue, a device should enforce that "Secure Connections" is supported on BT and BLE before running CTKD and raise an error if this is not the case.

CTKD Notifications CTKD is transparent to end-users and is specified in the standard as an optional feature. We exploit those facts to improve the stealthiness of our attacks. Given that CTKD is a security-critical feature we believe that it should not be considered optional, and a device should notify the user every time the feature is used. For example, the device should notify the user when she is re-pairing with a trusted device and is using CTKD to overwrite a long term key.

7 Related Work

Bluetooth standard compliant attacks are particularly dangerous as all Bluetooth devices are affected, regardless of version numbers or implementation details. Such standard-compliant attacks have appeared since the first versions of Bluetooth [24, 29]. Standard-compliant attacks on BT include attacks on legacy pairing [37], secure simple pairing (SSP) [10, 21, 38], Bluetooth association [22], key negotiation [1], and authentication procedures [3, 28, 40]. Standard-compliant attacks on BLE include attacks on legacy pairing [35], key negotiation [4], SSP [10], and GATT [25] Compared to the mentioned attacks that target either BT or BLE, the BLUR attacks are the first standard-compliant attacks targeting the intersection between BT and BLE.

We have seen attacks targeting specific implementation flaws on BT [6] and BLE [7, 20]. As our BLUR attacks target the specification level, they are effective regardless of the implementation details. Several surveys on BT and BLE security were published [17, 31, 32] but none of those surveys (and the Bluetooth standard) is considering CTKD as a threat. We here demonstrate that CTKD is a serious threat and must be included in the threat model.

Cross-transport attacks were exploited for proximity technologies using Bluetooth and Wi-FI. Two prominent examples are attacks on Apple ZeroConf [9] and Google Nearby Connections [2]. Our BLUR attacks are the first cross-transport attacks for BT and BLE.

The cryptographic primitives used by Bluetooth have been extensively analyzed. For example, the E_0 cipher used by BT was investigated [19] and it is considered relatively weak [32]. SAFER+, used for authentication, was analyzed as well [27]. BT and BLE "Secure Connections" use the AES-CCM authenticated-encryption cipher. AES-CCM was extensively analyzed [26, 34] and it is FIPS compliant. Our BLUR attacks target key negotiation and not cryptographic primitives, and are effective even with perfectly secure cryptographic primitives.

8 Conclusion

In this work, we present the first security analyses of CTKD and identify cross-transport issues and attacks against BT and BLE. CTKD enables an attacker to cross the security boundary between BT and BLE. These wireless transports have different security architectures and threat models. Despite this fact, the Bluetooth standard does not include CTKD in the BT and BLE threat models and the security implications of CTKD are not well understood.

We identify five cross-transport issues related to roles, "Secure Connections", association, device states, and key overwrite. Using the issues, we design and implement novel cross-transport attacks against BT and BLE enabling impersonation, traffic manipulation, and malicious session establishment. Our standard-compliant attacks exploit BT and BLE just by targeting one of the two. We name our attacks BLUR attacks as they blur the security boundary between BT and BLE.

We provide and discuss a low-cost implementation of the BLUR attacks using off-the-shelf hardware and open-source software. To demonstrate that our attacks are practical, we use our implementation to successfully attack 13 devices from different hardware and software manufacturers. Our devices range across all the Bluetooth versions supporting CTKD (version greater or equal to 4.2) and also Bluetooth 4.1. As the BLUR attacks are standard-compliant, all devices supporting CTKD are potentially vulnerable.

We sketch a set of countermeasures to address the BLUR attack directly in the Bluetooth standard. The countermeasures require to keep additional state about paired devices. We have disclosed our findings and our countermeasures to the Bluetooth SIG in May 2020.

References

[1] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper 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, August 2019.

[2] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. Nearby Threats: Reversing, Analyzing, and Attacking Google's "Nearby Connections" on Android. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, February 2019.

[3] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. BIAS: Bluetooth Impersonation AttackS. In *Proceedings of Symposium on Security and Privacy (S&P)*. IEEE, May 2020.

[4] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. Key Negotiation Downgrade Attacks on Bluetooth and Bluetooth Low Energy. *Transactions on Privacy and Security (TOPS)*, 2020.

[5] AOSP. Fluoride Bluetooth stack. https://chromium.googlesource.com/aosp/platform/system/bt/+/master/README.md, Accessed: 2020-01-27, 2020.

[6] Armis Inc. The Attack Vector BlueBorne Exposes Almost Every Connected Device. https://armis.com/blueborne/, Accessed: 2018-01-26, 2017.

[7] Armis Inc. BLEEDINGBIT: The hidden attack surface within BLE chips. https://armis.com/bleedingbit/, Accessed: 2019-07-24, 2019.

[8] Python Cryptographic Authority. Python cryptography. https://cryptography.io/en/latest/, Accessed: 2019-02-04, 2019.

[9] Xiaolong Bai, Luyi Xing, Nan Zhang, XiaoFeng Wang, Xiaojing Liao, Tongxin Li, and Shi-Min Hu. Staying secure and unprepared: Understanding and mitigating the security risks of apple zeroconf. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 655–674. IEEE, 2016.

[10] Eli Biham and Lior Neumann. Breaking the bluetooth pairing–fixed coordinate invalid curve attack. http://www.cs.technion.ac.il/~biham/BT/bt-fixed-coordinate-invalid-curve-attack.pdf, Accessed: 2018-10-30, 2018.

[11] Philippe Biondi. Scapy: Packet crafting for python2 and python3. https://scapy.net/, Accessed: 2018-01-26, 2018.

[12] Bluetooth SIG. Bluetooth Core Specification v5.2. https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=478726, Accessed: 2020-01-27, 2019.

[13] Bluetooth SIG. Bluetooth Markets. https://www.bluetooth.com/markets/, Accessed: 2019-10-23, 2019.

[14] BlueZ. Bluetooth 4.2 features going to the 3.19 kernel release. http://www.bluez.org/bluetooth-4-2-features-going-to-the-3-19-kernel-relea Accessed: 2020-01-27, 2014.

[15] Cypress. BLE and Bluetooth. https://www.cypress.com/products/ble-bluetooth, Accessed: 2020-01-27, 2019.

[16] Cypress. CYW920819EVB-02 Evaluation Kit. https://www.cypress.com/documentation/development-kitsboards/cyw920819evb-02-evaluation-kit, Accessed: 2019-11-16, 2019.

[17] John Dunning. Taming the blue beast: A survey of bluetooth based threats. *IEEE Security & Privacy*, 8(2):20–27, 2010.

[18] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38B.pdf, 2018. Recommendations of the NIST.

[19] Scott Fluhrer and Stefan Lucks. Analysis of the E0 encryption system. In *Proceedings of the International Workshop on Selected Areas in Cryptography*, pages 38–48. Springer, 2001.

[20] Garbelini, Matheus and Chattopadhyay, Sudipta and Wang, Chundong. SweynTooth: Unleashing Mayhem over Bluetooth Low Energy. https://asset-group.github.io/disclosures/sweyntooth/sweyntooth.pdf, Accessed: 2020-04-08, 2020.

[21] Keijo Haataja and Pekka Toivanen. Two practical man-in-the-middle attacks on Bluetooth secure simple pairing

and countermeasures. *Transactions on Wireless Communications*, 9(1):384–392, 2010.

[22] Konstantin Hypponen and Keijo MJ Haataja. "nino" man-in-the-middle attack on bluetooth secure simple pairing. In *Proceedings of the International Conference in Central Asia on Internet*, pages 1–5. IEEE, 2007.

[23] Intel. Intel Wireless Solutions. https://www.intel.com/content/www/us/en/products/wireless.html, Accessed: 2020-01-27, 2019.

[24] Markus Jakobsson and Susanne Wetzel. Security weaknesses in Bluetooth. In *Proceedings of the Cryptographers' Track at the RSA Conference*, pages 176–191. Springer, 2001.

[25] Sławomir Jasek. Gattacking bluetooth smart devices. Black Hat USA Conference, 2016.

[26] Jakob Jonsson. On the security of CTR+ CBC-MAC. In *Proceedings of the International Workshop on Selected Areas in Cryptography*, pages 76–93. Springer, 2002.

[27] John Kelsey, Bruce Schneier, and David Wagner. Key schedule weaknesses in SAFER+. In *Proceedings of the Advanced Encryption Standard Candidate Conference*, pages 155–167. NIST, 1999.

[28] Albert Levi, Erhan Çetintaş, Murat Aydos, Çetin Kaya Koç, and M Ufuk Çağlayan. Relay attacks on Bluetooth authentication and solutions. In *Proceedings International Symposium on Computer and Information Sciences*, pages 278–288. Springer, 2004.

[29] Andrew Y Lindell. Attacks on the pairing protocol of Bluetooth v2.1. *Black Hat USA, Las Vegas, Nevada*, 2008.

[30] Dennis Mantz, Jiska Classen, Matthias Schulz, and Matthias Hollick. InternalBlue - Bluetooth binary patching and experimentation framework. In *Proceedings of Conference on Mobile Systems, Applications and Services (MobiSys)*. ACM, June 2019.

[31] Nateq Be-Nazir Ibn Minar and Mohammed Tarique. Bluetooth security threats and solutions: a survey. *International Journal of Distributed and Parallel Systems*, 3(1):127, 2012.

[32] John Padgette. Guide to bluetooth security. *NIST Special Publication*, 800:121, 2017.

[33] Qualcomm. Expand the potential of Bluetooth. https://www.qualcomm.com/products/bluetooth, Accessed: 2020-01-27, 2019.

[34] Phillip Rogaway. Evaluation of some blockcipher modes of operation. *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan*, 2011.

[35] Mike Ryan. Bluetooth: With low energy comes low security. In *Proceedings of USENIX Workshop on Offensive Technologies (WOOT)*, volume 13, pages 4–4. USENIX, 2013.

[36] Mike Ryan. Pybt: Hackable bluetooth stack in python. https://github.com/mikeryan/PyBT, Accessed: 2019-06-19, 2015.

[37] Yaniv Shaked and Avishai Wool. Cracking the Bluetooth PIN. In *Proceedings of the conference on Mobile systems, applications, and services (MobiSys)*, pages 39–50. ACM, 2005.

[38] Da-Zhi Sun, Yi Mu, and Willy Susilo. Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard v5. 0 and its countermeasure. *Personal and Ubiquitous Computing*, 22(1):55–67, 2018.

[39] National Security Agency USA. Ghidra: A software reverse engineering (SRE) suite of tools developed by NSA's research directorate in support of the cybersecurity mission. https://ghidra-sre.org/, Accessed: 2019-02-04, 2019.

[40] Ford-Long Wong, Frank Stajano, and Jolyon Clulow. Repairing the Bluetooth pairing protocol. In *Proceedings of International Workshop on Security Protocols*, pages 31–45. Springer, 2005.

[41] Joshua Wright. I Can Hear You Now - Eavesdropping on Bluetooth Headsets. https://www.willhackforsushi.com/presentations/icanhearyounow-sansns2007.pdf, Accessed: 2018-10-30.

[42] Apple WWDC. What's New in Core Bluetooth. https://developer.apple.com/videos/play/wwdc2019/901, Accessed: 2020-01-27, 2019.