Method Confusion Attack on Bluetooth Pairing

Maximilian von Tschirschnitz, Ludwig Peuckert, Fabian Franzen, and Jens Grossklags
Technical University of Munich
Email: {maximilian.tschirschnitz,ludwig.peuckert}@tum.de, franzen@sec.in.tum.de, jens.grossklags@in.tum.de

***Abstract*—Bluetooth provides encryption, authentication, and integrity protection of its connections. These protection mechanisms require that Bluetooth devices initially establish trust on first use through a process called pairing. Throughout this process, multiple alternative pairing methods are supported.**

In this paper, we describe a design flaw in the pairing mechanism of Bluetooth. This flaw permits two devices to perform pairing using differing methods. While successfully interacting with each other, the devices are not aware of the Method Confusion. We explain how an attacker can cause and abuse this Method Confusion to mount a Method Confusion Attack. In contrast to other attacks targeting the pairing method, our attack applies even in Bluetooth's highest security mode and cannot be mitigated in the protocol. Through the Method Confusion Attack, an adversary can infiltrate the secured connection between the victims and intercept all traffic.

Our attack is successful in practically relevant scenarios. We implemented it as an end-to-end Proof of Concept for Bluetooth Low Energy and tested it with off-the-shelf smartphones, a smartwatch and a banking device. Furthermore, we performed a user study where none of the 40 participants noticed the ongoing attack, and 37 (92.5%) of the users completed the pairing process. Finally, we propose changes to the Bluetooth specification that immunize it against our attack.

I. Introduction

Bluetooth has steadily gained prominence as a communication protocol for wireless, short-distance, device-to-device communication. In 2010, the Bluetooth Special Interest Group (Bluetooth SIG) standardized Bluetooth Low Energy (BLE), offering a low-cost and low-power communication protocol to IoT vendors [1]. Further enhancing its popularity, BLE is now used in a wide range of products in mobile computing, healthcare, finance, energy, logistics and entertainment applications. In 2019, the Bluetooth SIG expected that 4 billion Bluetooth capable devices would be shipped by the end of the year; about 3.2 billion with support for BLE [2]. Many of these devices handle sensitive data or run critical applications and therefore require heightened security to protect communications. For example, smartwatches or fitness trackers often connect to their owner's smartphone via BLE to exchange personal user data and notifications.

There are known attacks on Bluetooth Classic (BC) and BLE including downgrade or cryptographic attacks [3, 4, 5, 6, 7]. However, none of them apply to the current Bluetooth 5.2 specification if a secure connection mode is used.

To account for the need for security, Bluetooth offers encryption, authentication, and integrity protection on application request. In order to support these features, a trusted connection has to be established between the participating devices. This procedure, commonly known as *pairing process*, establishes trust on first use.

In Bluetooth, multiple pairing methods are available. Therefore, both devices need to mutually agree on one of these methods. Depending on the pairing method, the user may be involved and obligated to transfer authentication values between the devices to authenticate the pairing.

However, some pairing methods pick those authentication values from the same value space. Furthermore, the Bluetooth pairing fails to verify whether both devices actually conduct the same pairing method. It is, therefore, possible that two separate pairing processes which perform entirely different pairing methods interact with each other. Even though the user participates in the pairing process, she is not provided enough information to recognize such a *Method Confusion*.

In this paper, we show that an attacker can abuse this flaw and attack the pairing process by applying an adversarial action we call *Method Confusion Attack*. The attacker primarily intercepts and hijacks the pairing attempt between two devices (which have not yet established a trust connection). Subsequently, the attacker performs two different pairing methods with both victims (Method Confusion). The victims at this point assume to be pairing with their desired peer. The attacker now gains secret information, which in turn can be used to influence the pairing processes in such a way that the Method Confusion concludes in successful pairings. While the victims assume to have established a trusted connection, they instead paired with the attacker, who is now in a stable Man-in-the-Middle (MitM) position. The cryptography of each single pairing method itself is not broken by our attack. It cannot be prevented by Bluetooth's existing security mechanisms. Changes to the specification are required to mitigate it.

Our contributions can be summarized as follows:

- We introduce the Method Confusion Attack. It abuses a design flaw to establish a MitM position in a specification-compliant 'secure' setup consisting of two BLE devices.
- We show how our attack impacts millions of device combinations by testing the vulnerability of popular off-the-shelf devices (smartwatch, smartphone, banking device).
- We discuss that in certain implementations our attack could be spotted and mitigated by an informed and careful user. We conducted a user study in which none of the 40 participants noticed the attack.
- Based on these findings, we propose an implementation hotfix for device vendors and also suggest multiple long-term fixes for the Bluetooth specification.

Maximilian von Tschirschnitz and Ludwig Peuckert are co-first authors.

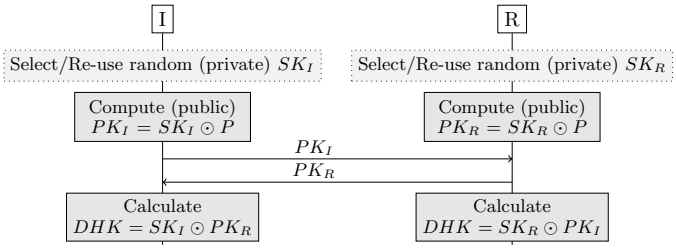


Fig. 1. ECDH Public Key Exchange and schematic calculation of shared secret.

In this work, we describe how our attack targets BLE devices. While our arguments and experiments will be based solely on BLE, we also discuss how this attack is applicable to BC (see Section VIII-C).

We disclosed this design flaw to the Bluetooth SIG. They acknowledged our findings and are cooperating with us to target the issue.

II. Bluetooth Background

As a primer for the discussion of the attack and related work, we now provide a short summary of important aspects of BLE.

Introduced in 2010 by the Bluetooth SIG, BLE is included in every iteration of the Bluetooth specification since version 4.0. Since then, its popularity and market share have been rapidly growing, replacing BC in many new devices [2].

BLE can provide an encrypted, integrity-protected, and authenticated connection between devices, if this is requested by the application. The basis for these security features is established in a process called pairing. First, the user has to enable Bluetooth advertisements, making the device visible to other Bluetooth devices. Next, the pairing process is initiated on the other device. At the end of pairing, both devices have authenticated each other and share a key (e.g., the *LinkKey*) for cryptographically secure data transfer.

Since its creation, the Bluetooth specification was amended multiple times with new pairing methods. In the following, we will discuss methods that concern BLE. Later in Section VIII-C, we will point out how the pairing methods and vulnerabilities of BLE correspond and likely translate to the ones of BC.

- *Legacy Pairing* is supported in BLE since its introduction in version 4.0. It acts by establishing a shared secret between both peers. This is achieved by either entering the same PIN on both devices, Out of Band (OOB) (e.g., NFC) or via a preconfigured publicly known value (e.g., Unit Key or Just Works). The established secret is then used to derive the necessary encryption keys. There exist simple attack schemes for this form of pairing (*cf.* Section IX-A). Connections that rely on this method can, therefore, not be considered as significantly protected. Hence, we will not describe this method any further.
- *Low Energy Secure Connections* is supported by BLE since version 4.2. Low Energy Secure Connections (LESC) uses an Elliptic Curve Diffie-Hellman (ECDH)-based key exchange on curve P-256 [1, Vol. 3 Part H 2.3.1] to negotiate a long-term secret. This is a significant improvement over the open key exchange of legacy pairing since an adversary cannot acquire the key by passive eavesdropping. Further, ECDH supports perfect forward-secrecy for the established connection.
  If protection against an active MitM attacker is required, the origin of the key exchange material is verified by a 'secure backchannel'. For instance, the user might act as this backchannel. The verification is performed through one of four *Association Models*; chosen dependent on device capabilities and security requirements.

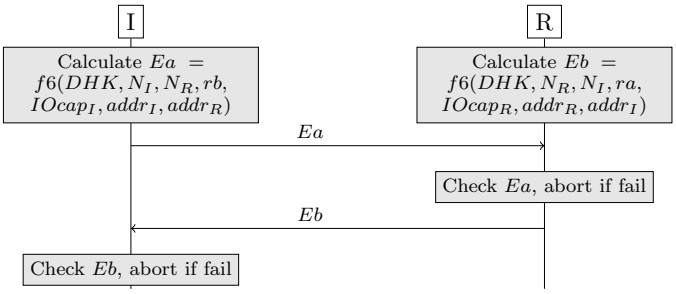


Fig. 2. Secure Connections LTK validation. $iocap_X$: IOCaps of X; $addr_X$: X's device address; ra, rb, N_X: values exchanged in Authentication stage (cf. Fig. 3,4).

LESC is the method mainly targeted by our attack, we will, therefore, expand upon it in the following.

A. The Pairing Process

In the following, we describe LESC-based pairing, as it is defined in the specification [1, Vol. 3 Part H 2.3]. The device initiating the pairing process is referred to as *Initiator* (I), the answering device is called the *Responder* (R).

1) Pairing Feature Exchange: Before the actual pairing process, both devices exchange information about their respective security requirements and Input-Output Capabilities (IOCap)s.

2) Public Key Exchange: Both devices exchange their ECDH Public Key (PK) information and calculate their shared Diffie-Hellman (DH) key. The exchange is visualized in the schemata of **Fig. 1**. The $\odot$ operation is to be interpreted as scalar multiplication on the Elliptic Curve (EC) body.

Please note that the parties have not yet authenticated each other's PKs (PK_I and PK_R). Therefore, the shared secret (DHK) established using those PKs cannot be trusted.

3) Authentication: Only in the subsequent Authentication stage are these previously exchanged PKs actually authenticated. The method used for authentication (i.e., the Association Model) varies. The decision on which Association Model will be used is based on the information gathered in the previously mentioned Pairing Feature Exchange. Both devices conduct this model-selection process independently. After completion of this decision process, both devices assume that their pairing partner has concluded upon the same method as they did. The exact decision process is detailed in Section II-C.

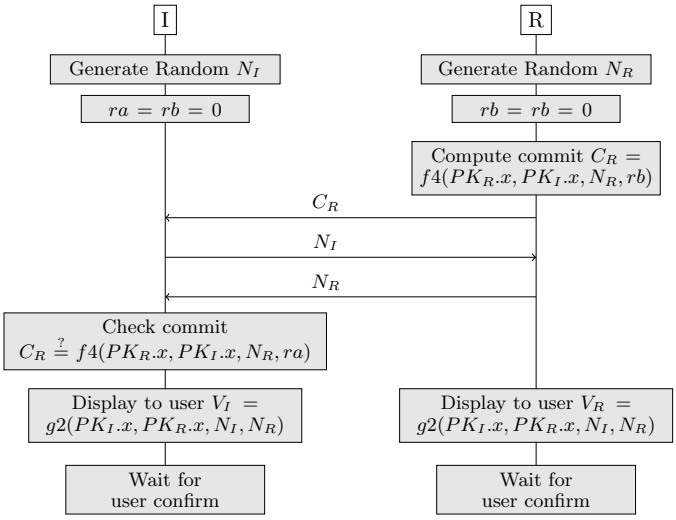


Fig. 3. Numeric Comparison.

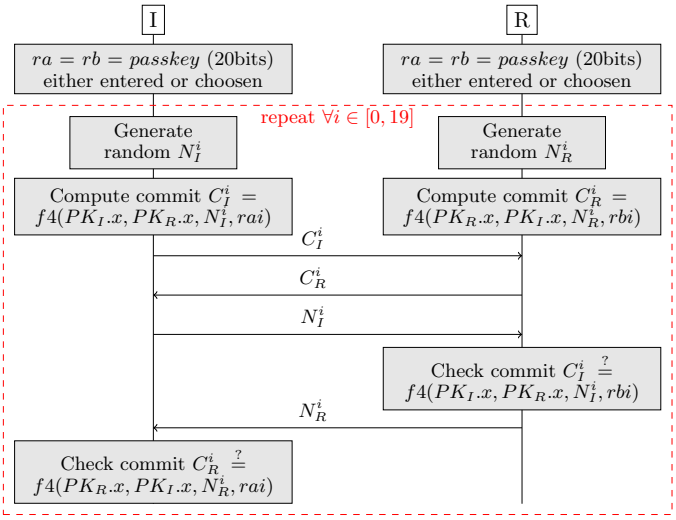


Fig. 4. Passkey Entry.

The devices can conclude on one of the following Association Models:

- *Just Works:* The keys are not authenticated (i.e., *unauthenticated* security requirement).
- *Out of Band:* The PKs are authenticated over a backchannel separate from Bluetooth (e.g., NFC, QR-Code).
- *Numeric Comparison:* The user is displayed a 6-digit number on both devices and has to confirm if they are equal.
- *Passkey Entry:* The user is displayed a 6-digit passkey on one device and is asked to enter it into the other device.

Only the Numeric Comparison (NC) and Passkey Entry (PE) methods (besides OOB) are capable of properly authenticating the peer and avoiding MitM attacks as discussed in Section IX-C (MitM protection). NC and PE are central to our attack and are discussed in further detail in Section II-B1 and II-B2.

After completion of the Authentication stage, the integrity and authenticity of the PKs are assumed to be verified.

Note that *no stage* does in fact authenticate whether both partners have performed the same Association Model. This is a design flaw, which enables our Method Confusion Attack (as acknowledged by Bluetooth SIG; see Section X).

4) Long-Term Key Calculation and Validation: The now trusted PKs are then used to establish a secure channel between both parties. The LTK and $MacKey$ are derived from the now authenticated PKs:

$$MacKey||LTK = f5(DHK, N_I, N_R, addr_I, addr_R) \quad (1)$$

where $f5$ is a cryptographic key generation function described in the Bluetooth specification [1, Vol. 3 Part H 2.2.7]. N_I and N_R are Nonces that were exchanged in the Authentication stage (e.g., Section II-B1). The $MacKey$ is of no further importance for our attack.

Subsequently, confirmation values are calculated on both sides (Ea and Eb). They are then exchanged over the new secure channel and are validated by the peers as can be seen in **Fig. 2**. Based on the trust established with the peers' PKs, this step retrospectively validates integrity and authenticity of all previously exchanged values.

B. MitM protected Association Models

Besides OOB, NC and PE are the only Association Models which provide MitM protection. As our attack targets this kind of protected authentication, they are of central interest to us and are detailed in the following.

1) Numeric Comparison: NC authenticates the PKs by creating a hashsum of the PKs that are to be validated. This hashsum is presented in the form of a 6-digit number to the user on both devices. If the user confirms those numbers to be equal on both devices, the keys are authenticated.

To avoid replay attacks, this hash needs to involve random nonces (N_I and N_R). Those nonces get exchanged in a particular order as shown in **Fig. 3** and are preceded by a confirmation message. The confirmation message and the order of the exchanges are important. In particular, the confirmation message assures that no partner knows the other partner's nonce before they have fixed their own choice of nonce. Otherwise, an attacker would be able to easily find and choose nonces that cause collisions in the confirmation value Va, increasing the likelihood of wrongfully approving keys that are not legitimate.

2) Passkey Entry: The PE requires one of the devices to choose and display a random 6-digit passkey. This passkey has to be entered by the user into the other device. The device which displays is chosen based on its IOCap.

The authentication then gets executed as seen in **Fig. 4**. The steps in the marked box are executed for each of the 20 bits of the entered passkeys ($rai, rbi : i \in [0, 19]$). If the process does not abort before completion, the PKs are considered authenticated. This bit-wise challenge-response process assures that both parties are in possession of the correct passkey.

C. Association Model Agreement

Each device has individual capabilities and security requirements. For instance, laptops have a display and a keyboard, but

headphones might have neither. To support a large variety of devices, the Association Model is dynamically selected based on the pairing devices' capabilities and security requirements. In BLE, the Pairing Feature Exchange is used to agree on a pairing method. For the Association Model agreement, three features are of interest [1, Vol.3 Part H 2.3]:

- *OOB-bit:* Indicates that OOB data is ready
- *MitM-bit:* Indicates the requirement of authentication
- *IOCaps:* Indicate provided capabilities for user interaction

In BLE, the possible IOCaps are the following:

- *DisplayOnly:* The device can only display a 6-digit numeric value.
- *DisplayYesNo:* The device can display a 6-digit numeric value and the user can input a confirmation (yes or no).
- *KeyboardOnly:* The user can input a 6-digit numeric value and a confirmation.
- *KeyboardDisplay:* The device can display a 6-digit numeric value and the user can enter a 6-digit numeric value and a confirmation.
- *NoInputNoOutput:* The device has no ability to communicate with the user.

When the pairing features have been exchanged, the devices decide independently from each other which Association Method is used. First, if *any device* has OOB data of the peer ready, the OOB-bit is set and the OOB authentication method is used. Our attack is not applicable to OOB and we will not further discuss this method. Next, if no device sets the MitM-bit, Just Works (JW) is used as authentication. JW offers no MitM protection during pairing and therefore is not further discussed (attacks targeting JW are listed in Section IX-C).

If the MitM-bit is set, the IOCaps are used to determine whether NC or PE is used. **Fig. 5** shows the mapping of IOCaps to the resulting authentication method for Initiator and Responder, respectively. Based on this table some requirements can be derived. For instance, NC requires *DisplayYesNo* or *DisplayKeyboard* on both devices. PE requires *Display** at one device, and **Keyboard* at the other.

Furthermore, this description shows that it is possible to enforce a pairing method. For example, to perform NC with a *KeyboardDisplay* device, *DisplayYesNo* may be announced.

D. Advertising and Discoverability

As our Proof of Concept (PoC) implements the interception of advertisement packets, we also discuss basic features of the advertising process [1, Vol. 3 Part C 9.2]. Advertising in BLE is controlled by the Generic Access Profile (GAP). A BLE device that is ready to accept pairing requests will enter discoverable mode. The device will then periodically transmit advertising packets over one of three channels (37, 38, 39) [1, Vol. 4 Part E 7.8.5].

The interval of these transmissions has a fixed component of between 20 ms to 10.24 s [1, Vol. 4 Part E 7.8.5] and a small random component. This form of dynamic interval is chosen to reduce the chances of collisions on the medium.

		Initiator				
		Display Only	Display YesNo	Keyboard Only	NoInput NoOutput	Keyboard Display
Responder	Display Only	Just Works	Just Works	Passkey Entry ●	Just Works	Passkey Entry ●
	Display YesNo	Just Works	Numeric Comparison	Passkey Entry ●	Just Works	Numeric Comparison
	Keyboard Only	Passkey Entry ●	Passkey Entry ●	Passkey Entry ●	Just Works	Passkey Entry ●
	NoInput NoOutput	Just Works	Just Works	Just Works	Just Works	Just Works
	Keyboard Display	Passkey Entry ●	Numeric Comparison	Passkey Entry ●	Just Works	Numeric Comparison

● Responder displays, Initiator inputs
● Initiator displays, Responder inputs
● Initiator inputs and Responder inputs

Fig. 5. Provided IOCap and resulting Association Model for LESC [1, Vol. 3 Part H 2.3.5.1].

There are various types of advertising messages. For our scenario, the pairing with a yet unknown peer is most relevant. Therefore, we will focus on the advertisement message *ADV_IND*. This advertisement type signals surrounding devices that incoming connection requests (e.g., pairing) are accepted.

The advertisement message starts with the device address, which can have the following types [1, Vol. 3 Part C 15.1]:

- *Public Address:* Globally fixed; registered with IEEE.
- *Random Static Address:* Fixed and chosen randomly; does not usually change.

In case *LE privacy* is enabled, these types can also be used [1, Vol. 3 Part C 10.7]:

- *Private Resolvable Address:* Derived from a common secret (Identity Resolving Key (IRK)) between bonded partners; can change anytime.
- *Private Non-Resolvable Address:* Randomly generated; has not to be persistent.

Different data fields may follow the address in an arbitrary order. If the device is in discoverable mode, it includes also its *Local Name* [1, Vol. 3 Part C 9.2.3.2]. The Local Name is also the identifier presented to the user in the results of a device discovery ('Available Devices').

III. Method Confusion Attack

The Method Confusion Attack targets the pairing attempt of two BLE devices with the goal to achieve a MitM position. Instead of a single pairing between R and I, two pairings are conducted simultaneously with attacker M. I connects to the MitM Responder (M_R), and the MitM Initiator (M_I) connects to R. One of the pairings is performed via NC, while the other pairing is performed via PE. This leads to a situation virtually identical to a valid PE pairing between I and R (Method Confusion): One device displays a 6-digit value, and the other device prompts for a 6-digit value. However, the attacker has gained knowledge over the displayed value, which is then used to complete pairing with both victims.

This attack is mainly possible for three reasons:

1) The Association Models NC and PE use the same form of check value; i.e., it is not determinable whether a

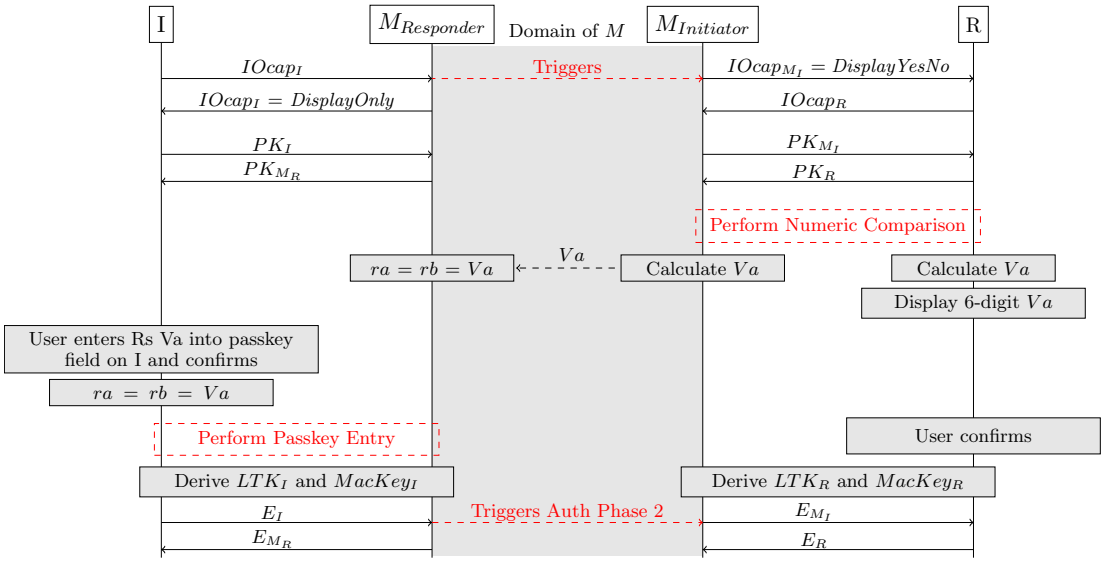


Fig. 6. Passkey on Numeric attack implementation. E_X is the confirmation value as calculated by device X.

given check value was generated by the NC or PE pairing process.

2) Devices are not authenticating which Association Model is actually used by their respective peers.
3) The specification does not prescribe any notification or wording that makes the user aware of the Association Model used. This makes it virtually impossible to recognize an attack; especially since the risk of a Method Confusion is not known to the user (*cf.* Section VI-A).

A. Attack Preparation

1) Initiator Connection to MitM: In order to apply Method Confusion, I must initiate pairing with M_R instead of R, so the attacker can act as MitM. We assume that the user attempts to pair two devices, I and R, with each other. R starts to advertise itself, while I searches for advertisements.

At the same time, M_R is also advertising as Responder under the same name as R. The user now observes M_R in the pairing menu of I, where M_R appears indistinguishable from R. To prevent the device R from showing up in the menu as well, R's advertisement signal may be jammed as we describe in our end-to-end PoC (*cf.* Section IV). Eventually, the user engages in pairing with M_R instead of R as they perceive M_R as the desired pairing partner.

2) MitM Interaction during Attack: As soon as the pairing request arrives at M_R, M_I initiates a connection with device R. Since I and R are now only communicating to M (M_R, M_I), M holds a MitM position during the pairing process. Note, that all the communication between M and I is handled by the M_R entity and all the communication between M and R is handled by the M_I entity. Both entities of M share their information with each other, and have access to each other's variables and state.

Based on the provided IOCaps of the attacked devices, our attack has two variations: *Passkey on Numeric (PoN)* and *Numeric on Passkey (NoP)*. Which of the two variants is applicable in each scenario is discussed in Section VIII-A.

We will now discuss the attack in detail.

B. Passkey on Numeric

In PoN, I performs PE with M_R, while R performs NC with M_I. **Fig. 6** shows the concurrent interactions of I and R with the MitM. First, the Pairing Feature Exchange between I and M_R is performed:

1) I initiates M_R and transmits its security requirements and IOCaps (*Keyboard**).
2) M_R responds to I and transmits its security requirements (set MitM-bit) and IOCaps (*DisplayOnly*).

This triggers M_I to begin Pairing Feature Exchange with R:

1) M_I initiates R and transmits its security requirements (set MitM-bit) and IOCaps (*DisplayYesNo*).
2) R responds to M_I and transmits its IOCaps (*DisplayYesNo* / *DisplayKeyboard*).

Next, PKs are exchanged between I and M_R respectively M_I and R simultaneously.

At the beginning of the authentication phase, M_R suspends the pairing procedure with I. In the meantime, M_I performs a NC-based authentication with the victim R:

1) C_R, N_I and N_R are exchanged.
2) M_I and R calculate Va.

Va is then presented to the user as a 6-digit number on R's display. R is now waiting for the user to compare the number and then to confirm the NC authentication. M_I has gained knowledge over Va.

Then, the PE procedure between M_R and I is continued:

1) M_R sets its passkey rb to Va
2) I and M_R gradually exchange passkey bits (ra_i).

At this point, I is requesting the user to enter a 6-digit passkey to perform the PE authentication.

In summary, I prompts the user to enter a 6-digit value while R displays a 6-digit value and awaits confirmation. This situation is virtually equivalent to a legitimate PE pairing dialogue. We assume the user transfers the value from R to I and confirms the dialogues. Dependent on the dialogue design and previous experience a user may be able to detect the ongoing attack at this point. In Section VI, we analyze the detection risk and reason that it is unrealistic for a user to spot an ongoing attack. To substantiate that reasoning, we report the results from a user study (Section VII), which align with our prognosis.

M_R chooses rb:

$$\text{On } M_R\text{: } rb = Va \tag{2}$$

Since the passkey entered on I is the Va displayed on R, the following holds:

$$\text{On } I\text{: } ra = Va \tag{3}$$

From (2) and (3) it follows:

$$\text{Between } I \text{ and } M_R\text{: } ra = rb \tag{4}$$

Therefore, the PE procedure completes successfully and the Authentication stage ends on device I and M_R.

Subsequently, the Long-Term Key (LTK) Calculation and Validation stage (Section II-A4) begins by I transmitting its confirmation value E_I. Since I and M_R have exchanged their PKs earlier, they will calculate the same DH-key DHK_{I,M_R}. The same applies to M_I and device R which calculate $DHK_{M_I,R}$.

After receiving the E_I from I, M_R will trigger M_I to calculate E_{MI} as:

$$E_{M_I} = f6(DHK_{M_I,R}, N_I, N_R, rb, IOcap_I, addr_I, addr_R) \tag{5}$$

E_{M_I} is then sent to R.

When the user confirms the NC dialogue on device R, the device completes its Authentication stage and waits for a confirmation message. As R now receives E_{M_I}, it validates the value successfully and replies with E_R to M_I. Therefore, the pairing between R and M_I is successfully completed.

Upon receipt of E_R on M_I, the message is discarded and M triggers M_R to calculate E_{M_R} as

$$E_{M_R} = f6(DHK_{I,M_R}, N_R, N_I, ra, IOcap_R, addr_R, addr_I) \tag{6}$$

This value is then transmitted to I.

I now validates the received E_{M_R} successfully, which also completes the pairing between I and M_R.

Note that M only completes the LTK Calculation and Validation stage if both victims have successfully completed the Authentication stage. If one victim does not complete the Authentication stage, the pairing procedures are eventually terminated by a timeout. Through that, the attacker prevents a so-called *one-sided pairing*. This behavior may be altered as an alternative attack vector as described in Section III-D.

As a result, I and M_R establish the same LTK. The same holds for M_I and R. All future communication is encrypted using keys derived from those LTKs. Consequently, M is able to relay all communication between I and R by decrypting the received messages and forwarding them after re-encryption with the respective peer's LTK. Therefore, M is able to eavesdrop on the cleartext of all messages which are exchanged over the encrypted channel between I and R. I and R are not aware of the attacker's presence.

C. Numeric on Passkey

NoP is similar to PoN. The largest difference is that in NoP I and M_R perform NC, while R and M_I perform PE. In consequence, PoN and NoP also differ in their implementation of the LTK Generation and Validation stage.

M_R performs NC-based pairing with I until the point where Va is displayed at I and known to M_R. Simultaneously, M_I performs PE-based pairing with R until the start of the Authentication stage.

As soon as both pairing couples complete these steps, the Va calculated by M_R is set as passkey at M_I. Then, the PE between M_I and R is continued until completion.

When the user confirms the NC dialogue at I, the message Ea is sent to M_R where it is discarded. Then, M_I calculates its confirmation message E_{M_I}, which is sent to R after PE has completed. R checks the received confirmation message successfully, calculates and replies with Eb, thereby completing the pairing between M_I and R.

When M_I receives Eb it discards the message, whereupon M_R calculates E_{M_R} and sends it to I. I receives the confirmation message and successfully validates it, upon which also the pairing between I and M_R is completed.

As with PoN, this allows the MitM to relay, eavesdrop and manipulate all further communication between I and R.

D. Alternative One-sided Pairing

If a user enters and confirms the passkey but does not confirm the NC, our original description of the attack would abort through a timeout eventually. Sometimes, though, it is a sufficient goal for an attacker to establish a connection to one of the victim devices; for instance, if the attacker aims to extract information like a phone book, fitness tracking data, or attempts to modify the target bank account settings of a point-of-sales terminal. For such scenarios, the attack can be amended by not aborting the pairing. We call this *One-Sided Method Confusion Attack*.

IV. Implementation

To prove the real-world viability of the attack, we designed an end-to-end attack framework. The framework initializes all devices and orchestrates its three components: 1) a Method Confusion Attack implementation, 2) a jammer, and 3) in case LE Privacy is enabled, also an address sniffer.

A. BThack

We implemented the Method Confusion Attack in our Bluetooth MitM platform *BThack*. BThack is based on *BTstack* [8], a well-established BLE stack offered by Bluekitchen GmbH. BThack allows its applications to modify the communication in various ways.

We combined it with USB *Cambridge Silicon Radio, Ltd., Bluetooth Dongles* (CSR dongles), which operate in HCI mode. Since BTstack is a commonly used commercial open-source product, we assume it to be an industry-proven and specification-compliant implementation of BLE.

In order to reduce the chances of unintended side-effects, the codebase and control flow of BTstack were altered as little as possible. This also assures simplified maintenance in case of upstream updates. To gain complete control over the pairing process, BThack extends the state machines of BTstack with *callbacks*. A BThack application registers custom callbacks to interrupt the control flow of the state machine at certain points. By virtue of BTstack's single-threaded design, the stack then suspends execution and allows memory modification before eventually resuming. This simplifies, for example, the synchronization of our MitM Initiator and Responder.

B. BThack MitM Application

The BThack MitM application consists of two memory-independent processes, each containing its independent BLE stack. Each of the processes communicates with its individual USB-BLE device (CSR dongle) and with each other using Inter Process Communication (IPC). One of the processes takes the role of a Responder (R), while the other acts as Initiator (I). Through IPC, they are able to synchronize the state machines of both virtual devices and relay messages.

The PoN attack as seen in **Fig. 6** was implemented as follows. The first victim device (I) connects to our MitM Responder (M_R). This, in turn, triggers the MitM Initiator (M_I) to connect to the device originally intended by the victim (R). At the passkey generation of M_R, a callback is placed. When triggered, the callback waits for M_I and R to finish NC and sets Va as passkey in M_R. M_I waits for the signal of M_R that PE was successful before it eventually confirms NC. After pairing is completed, the IPC is used to forward the communication between I and R.
NoP was implemented analogously with exchanged roles of Initiator and Responder.

C. Selective Jamming

As preparation for the Method Confusion Attack, the victim Initiator I has to establish a connection to our attack device M_R instead of R. We, therefore, have to prevent R's advertisement messages from reaching I, for instance by jamming them. At the same time, we have to avoid jamming the other device's, or even our own (M_R's), advertisements. To assure that, we perform a selective jamming of advertisements.

To perform *Selective Jamming*, we have to identify advertisement packets on-air and selectively induce interference. An interference message needs to be sent before the packet has finished transmission. Therefore, a low latency implementation is required. To fulfill that requirement, we adapted the work of Cayre et al. [9], which uses a customized firmware [10] for the *nrf51* BLE chip [11]. The firmware abuses the address matching functionality of the nrf51 [11, 17.1.13]. The (original) address matching feature allows to configure a 4-byte pattern that, if observed, triggers the packet reception process. The designated use of this functionality is to filter incoming packets for their header address to provide selective reception. The custom firmware instead utilizes this feature to scan for an arbitrary pattern on-air. Further, it configures the radio to stop any packet reception after 0 bytes and switch into transmission mode, transmitting a dummy packet. The caused interferences of two simultaneous accesses to the medium then alters the payload and causes a checksum mismatch of the advertisement packet; this causes the packet to be discarded upon reception. All of these actions are performed in the radio's hardware without CPU involvement keeping the latency as low as possible.

Bluetooth scrambles payload data with a standardized pseudorandom sequence before transmission [1, Vol. 3 Part D 1.1.2.2]. The offset of our matching pattern has to be known to pre-calculate the scrambling for it. The matching pattern can, therefore, be any 4-byte value that has a predictable occurrence and position in the target's advertisement payload.

If LE Privacy is not enabled, the advertisement address is always fixed to the beginning of the payload. In that case, the matching pattern is the first 4 bytes of the advertisement address.

Otherwise, the advertisement address is variable and we use the Local Name of the device as matching pattern. While the packet layout is variable, it is typically consistent between devices of the same model / implementation. We can, therefore, analyze one advertisement packet of R to obtain the position of the Local Name or alternatively guess it. The matching pattern is then set to the first 4 bytes of the Local Name. In order to avoid jamming our own advertisements of M_R, we move the Local Name entry to a different position in their payloads by placing a data entry before it.

Our hardware platform uses three *micro:bit* development boards, each hosting a nrf51 radio chip, to observe and jam all three advertisement channels simultaneously.

D. Interaction of the Components

The address of the victim is typically static. In fact, multiple studies have shown that LE Privacy mode is seldomly used among peripheral devices [12, 13, 14]. In that case, the jammer is configured to intercept the packets based on R's address.

However, if LE Privacy is enabled, R's Local Name is used for jamming. In that case, the framework also initializes a sniffer with disabled CRC check to keep track of the victim Responder's address. It constantly scans for R as it comes online and advertises.

Simultaneously to the jamming, the BThack MitM application is started to advertise under R's Local Name (*cf.* Section IV-B).

As soon as the BThack MitM application reports a connection attempt (from I to M_R), the jammer is halted. Then, the current address of R is passed from the sniffer to the MitM application which then initiates a connection between M_I and R. The attack subsequently completes as described above (*cf.* Section IV-B).

V. Evaluation

We evaluated our end-to-end PoC in four steps. We conducted a pre-test to validate the jamming functionality isolated from the actual Method Confusion Attack. Then, we conducted a full end-to-end lab-test to prove that the Method Confusion Attack is capable of gaining a MitM position among custom devices. Subsequently, we showed that our implementation also works with off-the-shelf devices. Finally, we conducted a performance evaluation to measure throughput and delay. For all tests, we used our jamming implementation with three micro:bit devices from Section IV-C.

For the lab test, we implemented a basic BTstack application running on CSR dongles. The Responder dongle R, the Initiator dongle I, and the malicious device M were placed so that the three devices formed a triangle with a right angle at R; the distance between I and R was $2\,\mathrm{m}$. This way, high-quality reception for all devices was guaranteed. Through these steps, we were able to accommodate for the weak antenna setup of our jamming application (micro:bits). With a high-power transceiver and better antenna the distance and constellation of the devices would be more flexible.

A. Testing Jamming of Bluetooth Advertisements

We validated the jamming functionality by conducting a series of repeatable tests. For that, we placed R into discoverable mode and attempted pairing from R while our jamming system was active. We tested jamming based on the known device address (LE Privacy disabled) and also on its Local Name (LE Privacy enabled) for a timespan of 10 minutes each (repeated 3 times). In all of our tests *no* advertisement messages from R reached I.

B. End-to-End Lab Test

In our full end-to-end test, we confirmed that our implementation of the Method Confusion Attack is capable of compromising a BLE connection that was established using LESC.

Since the Method Confusion Attack has two variations, we test two scenarios. The only difference between both test cases are the IOCaps of R.

- *Scenario 1:* Responder's IOCaps were *DisplayYesNo*; Initiator's IOCaps were *DisplayKeyboard*; attack scheme was PoN
- *Scenario 2:* Responder's IOCaps were *KeyboardOnly*; Initiator's IOCaps were *DisplayKeyboard*; attack scheme was NoP

The experimental procedure was the following: R was placed into advertising mode and I scanned for available devices. When I detected R's name among the scan results, it attempted a connection to that device. When a connection was established, pairing was requested.

When one of the devices was displaying a 6-digit value, while the other prompted a passkey field, we transferred and confirmed the value (*cf.* Section VI-A). As soon as pairing resulted in an encrypted connection, secret information was exchanged over that channel.

In both scenarios, the attacker used the end-to-end attack framework (*cf.* Section IV) to target the pairing attempts.

Both scenario tests succeeded, meaning that the attacker was in both cases able to establish a paired MitM position. Further, the attacker was able to eavesdrop on the secret information exchanged between the victims. This was verified by comparing the eavesdropped data with the cleartext that was legitimately decrypted by the victims.

C. Production-Device Evaluation

We also evaluated the attack with openly available off-the-shelf BLE-capable devices to verify the attack in real-world scenarios. The tested devices are likely to be used by regular users and may pose, if compromised, an open attack surface and critical data leak.

Again, we tested both the NoP and the PoN variant. In both scenarios, the attacked Initiators were a *OnePlus 7 Pro* with *Android 9.0*, an *iPhone 11* with *iOS 13.4.1* and a *Thinkpad W540* with *Windows 10*, all providing *DisplayKeyboard*. The two attacked Responder devices were:

- *Samsung Galaxy Watch 42mm:* smartwatch running *Tizen Wearable OS 4.0*; IOCaps were *DisplayKeyboard*; pairs with a smartphone using NC; attack scheme was PoN
- *Reiner SCT tanJack Bluetooth:* wireless TAN-Generator, IOCaps were *KeyboardOnly*; pairs with a smartphone using PE; attack scheme was NoP

Otherwise, the experimental procedure was equivalent to the lab-test.

The attacker was able to complete pairing with the victim devices and achieve a MitM position in all tested cases. The successful attacks proved the viability of Method Confusion with openly available unmodified devices.

D. Performance Evaluation

After a successful Method Confusion Attack, the MitM acts as additional hop between the attacked devices. To estimate the effects on network performance, we measured throughput and Round Trip Time (RTT).

Typically, BLE applications communicate over an abstraction layer called Generic Attribute Profile (GATT) which can guarantee a reliable, ordered communication [1, Vol. 3 Part G 1.1]. We implemented a basic GATT-based application in which both devices constantly transmit packets of fixed size. Upon reception, the packets are validated and counted. To measure the RTT, the application periodically sends a probing message that is acknowledged upon reception. The time difference in transmission of the probe and reception of the response is the RTT.

We evaluated the connection without MitM as well as with MitM in place (**Fig. 7**). Overall, the connection quality achieved appears to be sufficient for most applications. For instance, a VoIP call requires $21\,\mathrm{kbps}$ (throughput) and should for optimal quality not exceed an (RTT) of $300\,\mathrm{ms}$ [15].

	Direct		with MitM	
Payload Size	Throughput	RTT	throughput	RTT
200 B	392 kbps	165 ms	384 kbps	350 ms
100 B	400 kbps	80 ms	380 kbps	273 ms
50 B	296 kbps	52 ms	280 kbps	110 ms

Fig. 7. Average Performance of a GATT Measurement Application.

VI. Role of the User in the Attack

A. User Model

In the course of a Method Confusion Attack, one of the victim devices displays a 6-digit number and expects the user to compare and confirm that number. Simultaneously, the other victim device expects the user to enter a 6-digit number.

The Bluetooth specification does not provide any rules for these interactions. In fact, the design and wording on how expectations are communicated to the user vary between implementations. In **Fig. 8**, we collected a number of pairing dialogues from popular off-the-shelf BLE products.

For example, in a pairing attempt that is under attack of Method Confusion, the user is shown **Fig. 8a** on Android device I and **Fig. 8c** on Android device R.

We assume that the user would transfer the value between the devices and confirm both dialogues.

B. Chances of Detection

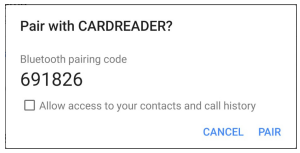
One may argue that under certain circumstances a user could realize that two different Association Models are performed. Users may spot this based on the expectations communicated to them.

We argue though, that the user would be typically unable to notice the Method Confusion Attack. Foremost, a regular user cannot be expected to know which actions a specification-compliant Bluetooth implementation may request from them. Furthermore, Method Confusion is based on a so far undisclosed design flaw in the pairing method; it was never intended that the user may need to verify consistent use of Association Models. Therefore, implementations do not specifically highlight the Association Model they employ.

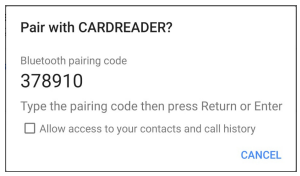
Our examples demonstrate, that it is difficult to distinguish the dialogue for NC and the dialogue for PE (on a display-device). More importantly, the difference between a legitimate and an attacked pairing attempt is sometimes even impossible to spot: When comparing the example attack from the above section, i.e., **Fig. 8a** and **Fig. 8c**, with a legitimate benign (i.e., not attacked) PE dialogue combination for Android (i.e., **Fig. 8b** and **Fig. 8c**), it is apparent that the user has only a negligible chance to realize any differences between **Fig. 8a** and **Fig. 8b**. Some interfaces (e.g. iOS) may be more intuitive and rather raise awareness with the user.

We have conducted a survey of over 35 popular devices (*cf.* Section VIII-A). None of these devices differentiated their PE-display dialogues significantly from their NC dialogues.

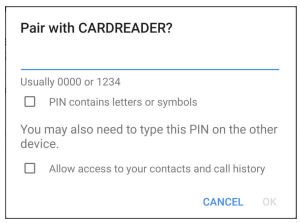
It is, therefore, unlikely that the user recognizes any irregularities. To further strengthen that assumption, we conducted a user study in which none of the 40 participants recognized the attack.



(a) Android 10.0 - Numeric Comparison.



(b) Android 10.0 - Passkey Display.



(c) Android 10.0 - Passkey Enter.

(d) TAN Generator - Passkey Enter (Translation: **"Enter Pairing-Code:"**)

(e) Samsung Galaxy Smart Watch - Numeric Comparison.

Fig. 8. Examples of common pairing dialogues.

VII. User Study

To further investigate the chances of attack detection, we performed a user study to determine whether participants recognize irregularities in the pairing dialogs when a Method Confusion Attack is performed.

A. Study Preliminaries

The study was conducted at the campus of a major research university, where participants were recruited at a public social event. Due to the location, a rather technically-experienced participant group was recruited. We argue that a higher educated and technical audience provides a stricter evaluation for our attack. Any individuals known to the experimenters like family members or friends were excluded from the study to avoid acquaintance bias. As compensation for their participation, participants were granted a food voucher.

Our institution does not require IRB approval for user studies, but we carefully designed our study to adhere to common standards in the security and usability community [16]. All participants signed a consent form and had to read and accept a privacy disclaimer before data collection was performed. To account for the priming effect, subjects were told that they were participating in a 'user study about wireless connectivity'. We offered a debriefing option to participants after conclusion of the study. The data collected in the study was safely stored and only accessible to the participating researchers. The appendix provides an overview of the participant demographics.

B. Experimental Procedure

The target of the study was to evaluate how users would behave when confronted with a Method Confusion Attack

on everyday devices. The subjects were provided with three common Bluetooth-capable devices (A–C) and a Bluetooth-enabled smartphone (D):

- *A. Wireless Headphones:* Sony WH-1000XM3; capable of OOB (NFC) pairing.
- *B. Wireless In-Ear-Headphones:* SoundPeats TrueCapsule; capable of JW pairing.
- *C. Mobile TAN Generator:* Reiner SCT tanJack Bluetooth; capable of PE pairing.
- *D. Android Smartphone:* Samsung Galaxy S8 Edge - Android 9.0; capable of all LESC pairing methods.

The devices were provided together with their vendor-supplied manuals.

The subjects were asked to pair the devices (A–C) with the smartphone (D). This was described as preparation to subsequently fill out a short survey about their user experience with those wireless devices (given in the Appendix). There was no time limit imposed. Instead, the participants were told that they were free to stop their attempts whenever they felt it suitable. For instance, if they grew frustrated with repeatedly failed pairing attempts or bad UI design.

After explanation of the procedure, the participants were left alone with the task. This way, we prevented that any cues from the experimenter would be "picked up" by the participants [17]. We addressed the "helpful participant" bias by two additional measures that were clearly communicated to all participants beforehand. It was urged that the main objective of the study would be to collect user experience, and that failed attempts had the same value to our research as successful ones. The participants were also made aware that they would receive the compensation either way; even if a pairing attempt did not succeed and they would move on to another device.

After they attempted the pairing, the survey was conducted as announced irrespective of whether they were able to pair with all the devices or not.

C. Hidden Attack

The users were not made aware of the fact that the pairing between devices C and D was attacked using Method Confusion. To strike a reasonable balance between simple study design and realism, we did not conduct the jamming process in the background, but simulated it. As preparation for that the devices C and D were once paired with each other in advance. After successful pairing, device C was showing up under the 'Paired Devices' section in the Bluetooth menu of D. C was not shown anymore under 'Available Devices' even if C was advertising. Finally, we renamed the device C in D's Bluetooth menu from its original name of 'tanJackXXXX' to the inconspicuous designator 'Audi Q6' (i.e., a well-known car manufacturer). Therefore, the user was now unable to find the original device when it began advertising.

Instead, the BThack MitM application advertised itself under the name 'Reiner SCT*', which was the device name printed prominently over the device's screen. If the user initiated the pairing to the attacker's MitM application, the attack would be performed as described in Section III-C.

		Initiator		
		DisplayYesNo	KeyboardOnly	KeyboardDisplay
Responder	Display YesNo	Attack not applicable	Numeric on Passkey	Numeric on Passkey
	Keyboard Only	Passkey on Numeric	Attack not applicable	Passkey on Numeric
	Keyboard Display	Passkey on Numeric	Numeric on Passkey	Both variants applicable

Fig. 9. Mapping victim IOCap to MitM IOCap

D. User Study Results

The study showed that 37 (92.5%) of the participants eventually entered the NC-value shown in device D into the PE field of device C. Through that action, the MitM attacker was able to complete pairing with both devices. In turn, 3 (7.5%) of the participants were not able to conduct the pairing. Since the situation presented to the participants was virtually identical to a valid PE pairing, none of the participants suspected an attack. Instead, all of the failed attempts were assumed to be their own mistakes or Bluetooth connectivity issues.

In conclusion, the study lends substantial support to our argumentation in Section VI-A.

VIII. Discussion

In this subsection, we present data on the applicability of the Method Confusion Attack to the Bluetooth device market. Next, we discuss possible short-term and long-term fixes. Finally, we provide information on the applicability of the attack for BT Classic.

A. Applicability, Restrictions, and Impact

As a reminder, we summarized the attack requirements in the matrix in **Fig. 9**. Further, we already successfully demonstrated our attack with several popular devices (see Section V). While many attacks exist which target the Bluetooth pairing process (*cf.* Section IX) none of them are applicable when MitM protection is used. Relevant organizations in the standards space like NIST officially strongly recommend that all vendors that use Bluetooth should utilize the strongest Bluetooth security modes, as those enable MitM protection [18]. Our attack specifically targets devices that rely on that protection feature.

While MitM protection may not be common for the average Bluetooth device, security critical device classes are enforcing MitM protection more regularly.

The Bluetooth SIG's Launch Studio lists 22,757 Declaration IDs for "previously qualified designs and declared products" for the time period of January 2015 - April 2020 [19]. Of these, 17,154 are specifically for end products including many devices from popular brands with large numbers of users. Further, many of the IDs are bulk listings for dozens or even hundreds of different products. For example, ID D049484 (February 27, 2020) contains 592 Toyota car models and details about the associated car multimedia system. While

the *public* part of Launch Studio does not provide a detailed enough search functionality to determine which devices actually rely on MitM protection, it allows to identify potentially sensitive device categories. For example, one can quickly identify devices that likely communicate highly sensitive information such as heart rate/pressure monitors, blood glucose monitors, or smart baby monitors. Access for SIG members even includes data that indicates support of MitM protection, but also does not state whether it is actually employed. Internet search (for manuals and setup videos) can then help to verify if a device actually uses MitM protection.

We investigated two sensitive device classes to better understand the distribution of MitM-protected devices. First, we researched which devices of their class are making up the relevant part of their market. Then, we determined whether these devices rely on the MitM protection of Bluetooth pairing.

1) Smartwatches: These devices are using Bluetooth to transfer notifications, messages, health data and location-tracking information to users' smartphones. For 2019, the overall number of shipped devices increased to 92.4 Million (up from 75.3 Million in 2018) [20]. "Major players" are Apple, Samsung, Fitbit, Garmin, and Fossil in a relatively "fragmented" global market [21]. We examined manuals and setup videos of these vendor's smartwatch device to determine their method of pairing (links to the manuals are provided in the Appendix). We found that all examined Samsung, Garmin and Fossil smartwatch devices are utilizing either NC or PE Bluetooth pairing, and are, therefore, enforcing Bluetooth's MitM protection. Apple watches perform OOB pairing and do, therefore, not rely on Bluetooth's MitM. Fitbit uses JW combined with application layer encryption. In summary, of the total market, at least 12.3% (first quarter 2018) – 15.1% (first quarter 2019), e.g., data for Samsung, Garmin, and Fossil, likely rely on Bluetooth's MitM protection [22].

2) Car Multimedia: Multimedia systems in cars interact with smartphones to provide hands-free telephony, GPS tracking and access control. We determined the best selling car models of 2019 [23]. Again, we examined setup manuals of the most recent entertainment system of these models (see Table in the Appendix) and determined that all of them are utilizing Bluetooth's MitM protection feature (i.e., either PE or NC for pairing). The 2019 sales of these cars amount to over 8.3 Million vehicles [23]. Note our Toyota example from above, which strongly suggests that the entire current range of products is using Bluetooth's MitM protection feature.

In summary, we assume that a sizable percentage of the markets for smartwatches and car entertainment systems is vulnerable to our attack.

B. Proposed Fix

In the following, we discuss possible countermeasures that can be implemented against the attack. For every fix, we discuss how it would affect the current device base and standards and how it influences other security aspects.

1) Enforcing Pairing Method: In some cases a vendor of a product can assume certain properties of the devices the product will be paired with.

This might be the case if the product is just to be paired with another device that is also issued by the vendor. In this case the pairing method can just be fixed by setting IOCaps on both products that restrict the Association Model to one specific method (*cf.* Section II-C).

2) User Interface Design Hotfix: The Bluetooth specification does not dictate a specific wording or UI design for user dialogues required for NC and PE. Therefore every device manufacturer and OS vendor chooses visual presentation and wording independently. Typically, these design decisions are dictated by device restrictions like screen size as well as quality and design premises like creating an intuitive uninterrupted user experience.

As a hotfix / first mitigation to the attack, vendors can warn the user against misusing the information presented. For instance, a NC dialogue box could warn the user of not entering the shown number anywhere, comparable to a credit card PIN. Colors and different text styles can be utilized to improve effectiveness. Uzun et al. [24] conducted a user study on the Bluetooth dialog in which they proved that a clearer wording can lead to drastically improved user awareness.

3) Authenticating Association Model: We argue that relying on the user to be aware of the Association Model is not a sustainable practice.

Primarily, the pairing methods of Bluetooth were designed with the goal to ease the pairing process and allow average users to conduct a secure pairing with minimal, simple interaction. The average user is typically also not aware that attacks like Method Confusion are a threat and special attention on the used Association Model is required. In fact, the only duty the user is made aware of is the task of comparing or entering numbers.

Secondly, a user often has none or no clear indication which Association Model is performed by the device and is not able to decide whether a pairing is under attack (*cf.* dialogue examples in Section VI-A).

We argue that Method Confusion needs to be prevented through the protocol itself. We suggest embedding the information on which Association Model is used into the information that is compared or transferred by the user. Specifically, we propose that a passkey used by PE has to be distinctly distinguishable to a value displayed in NC. For instance, a PE-passkey may always have its least significant bit set to 1 while a NC-value has its least significant bit set to 0. The application expecting a PE-passkey would therefore be able to detect and abort Method Confusion Attacks when it receives a NC value. This solution has the benefit of being fairly easy to amend to the existing protocol. Primarily, devices attempting to support this would not be required to implement new input methods. Secondly, the protocol would not require changes.

By using one bit of the 20-bit value to signal the Association Model, we reduce the space of possible values from $1.000.000$

		Initiator	
		DisplayYesNo	KeyboardOnly
Responder	Display YesNo	Attack not applicable	Numeric on Passkey
	Keyboard Only	Passkey on Numeric	Attack not applicable

Fig. 10. BC: Mapping victim IOCap to MitM IOCap

to 500.000. Therefore, a MitM attacker has better chances of correctly brute-forcing the passkey during pairing. However, given recent hardware capabilities this is still sufficiently large to prevent an attacker from guessing the correct PE-passkey / NC-value during the pairing process. Therefore, we argue that the benefits of this method may outweigh that drawback. As the Bluetooth SIG pointed out, another concern with this solution is its backward compatibility. Devices that do not follow these updated guidelines can cause the connection attempt to fail in 50% of the cases.

An alternative to reutilizing the existing value space is to extend and divide the existing one into two disjoint sets, each assigned to one of the Association Models. For instance, one may only allow 5 Latin letters for the PE-passkey and only 6 digits for the NC-value. While this may conserve or even reduce the guessing probability, we argue that this proposal is harder to amend. The implementation requires a significant change in protocol and device interfaces and affects backward compatibility significantly.

C. Applicability to BT Classic

While we have discussed our attack with a focus on BLE, we anticipated that it is also applicable to BC.

The commonly accepted secure methods of pairing in BC are Secure Simple Pairing (SSP) and Secure Connections (SC). SC is a mode of SSP that enforces higher requirements on the cryptographic primitives used. SSP itself is virtually equivalent to LESC with three exceptions:

- SSP uses the P-192 EC as default [1, Vol. 1 Part A 5.3]; only if operating in SC-only mode it uses P-256 (as LESC always does).
- SSP uses $E0$ [25] as encryption; only in SC-only mode it uses AES-CCM [26] (as LESC always does).
- SSP does *not* support KeyboardDisplay as IOCap.

Otherwise, the Bluetooth specification states that the four Association Models of SSP are functionally equivalent to the ones of LESC [1, Vol. 1 Part A 5.4.1]. We were made aware that KeyboardDisplay is not available in BC by the Bluetooth SIG during the disclosure process. Consequently, our attack matrix for BC is altered as displayed in **Fig. 10**. We argue, therefore, that our results are transferable between LESC, SSP, and consequently SC. The comments we received from the Bluetooth SIG during the disclosure process support this assumption (Section X).

IX. Related Work

To better contextualize how our attack relates to other attack schemes, we now present the most relevant ones that were discussed in prior work.

A. Offline PIN Crushing in Bluetooth Legacy Pairing

Jakobsson and Wetzel [3] describe how the legacy method of BC's key establishment is vulnerable to offline brute-force attacks on the used PIN. Essentially, an attacker may capture the communication of the key-generation material and the subsequent authentication handshake. This information is solely protected by the shared symmetric secret (i.e., the PIN).

The issue roots in the fact that the PIN is the only non-public information and is also limited in entropy (typically 6 digits). The attacker is, therefore, able to exhaust all possible PINs; by re-simulating the captured processes for each PIN. The attacker can terminate when the simulated authentication handshake was successful. This way, the used PIN can be found and the long-term secret can be calculated.

This attack was later refined by Kügler [27], who exploited the radio implementation of Bluetooth in combination with the aforementioned bruteforce attack to establish a MitM position between the victim devices.

BLE legacy pairing relies on similar methods to exchange the cryptographic key material and, therefore, suffers from similar vulnerabilities. Ryan [28] describes how these issues can be exploited in BLE by a passive attacker.

B. Passkey Entry Reuse Attack

PE is sometimes implemented with a device-specific but fixed passkey. This passkey is chosen randomly by the manufacturer and then printed on or stored in the device's ROM. Through that, the manufacturer aims to support the PE method on devices without display or with weak entropy sources.

This way of pairing disregards the Bluetooth SIG's advice to not re-use PINs. The attacks proposed by Lindell [4] and refined by Barnickel et al. [29] target these devices.

An attacker eavesdropping during the Authentication stage (*cf.* Section II-A3) may learn PK_I, PK_R and Ca_i, Cb_i, Na_i, and Nb_i (where i is the i-th bit of the passkey). Therefore, an attacker only needs to perform one hashing operation per passkey-bit to determine if a confirm message Ca_i / Cb_i was created with ra_i / rb_i set to 0 or 1. The whole passkey can be recovered in 20 hashing operations. The attacker then intercepts and aborts the pairing process after the Authentication stage has ended. If the same passkey is now re-used, the attacker just acts as MitM, pairing both devices as MitM using the now known passkey.

C. Just Works MitM Attack

In practice, many devices performing SSP, SC or LESC are not requesting MitM protection during pairing (*cf.* Section II-C). This is often the case if devices do not have the physical interfaces to facilitate MitM-protected pairing modes. Examples for such devices are In-Ear-Bluetooth headphones or screenless fitness trackers. According to the Association Model agreement process, JW is performed in these cases. Therefore, there is no protection against an active MitM attacker provided. An attacker may completely take over the communication with both parties and implement its PKs into the pairing process. Examples of such attacks can be found in the work of Hypponen and Haataja [5].

D. Downgrade Attacks

The exchange of IOCaps described in Section II-A1 is vulnerable to interception and tampering. This is due to the lack of authenticity between the two parties at this point of pairing. In consequence, an attacker can act as MitM falsifying the IOCap and, thus, downgrade the authentication method to JW [5]. Subsequently, an attacker may continue the attack as described in the previous Section IX-C. However, the MitM-bit protects against such an attack. We want to point out that the Method Confusion Attack is fundamentally different from a downgrade attack. Different from our proposal, downgrade attacks do not succeed if the MitM-bit is set.

E. Fixed Coordinate Invalid Curve Attack

Biham and Neumann [6] show that all SSP (and as they argue SC, LESC) pairing methods are vulnerable to a so-called *Fixed Coordinate Invalid Curve Attack*. The attack abuses that only the x-component of the PKs is validated during the Authentication stage. Therefore, the attacker can manipulate the y-value during the PK exchange. This way reducing the possible ECDH keyspace to two possible keys, which can then be guessed. The Bluetooth SIG addressed this vulnerability by amending the specification [30]. It now requires devices to validate whether the PKs lie on the ECDH-curve.

F. Attacks not Targeting the Pairing

While the attacks described in this section focus on the Bluetooth pairing process, there are also other attack vectors. For example, the KNOB attack [7] focuses on breaking the encryption itself by downgrading the entropy of a connection.

Fawaz et al. [31] found that many BLE devices do not properly implement privacy-preserving features, such as *device address randomization*. This allows adversaries to track users through advertisement messages of their BLE devices. The authors developed *BLE-Guardian*, which hides the presence of selected devices from an attacker.

Further, it is a conundrum that Bluetooth applications are challenging to implement correctly. This is mainly due to the huge size and complexity of the specification. Multiple attacks resulting from implementation mistakes were uncovered by careful analysis like [32] and [33].

X. Responsible Disclosure Process

We reached out to the Bluetooth SIG as well as to Google and Apple; the two vendors with the largest market share for mobile operating systems. We received immediate responses from all parties and were also contacted by CERT/CC which takes part in coordinating the efforts. Google and Apple informed us that they take the issue very seriously and are working on a solution, but gave no concrete details. The Bluetooth SIG pointed out that certain limitations of our attack exist for BC. Those comments were incorporated into this paper's revision. Otherwise, the SIG acknowledged our findings, i.e., the vulnerability of BLE, and agreed that the same weakness should also apply to BC. They did not approve to incorporate our proposed fixes into the specification:

> 'The Bluetooth SIG and the working group concur with your paper's findings that the proposed method does expose the end-user to a viable man-in-the-middle attack, however the recommended changes to mitigate this attack vector at the protocol level were not approved.' [excerpt from email 12.3.2020]

They justified this decision through the lack in backward compatibility of our proposed fixes. Instead, the SIG announced the decision to reach out to their members about the issue with the help of CERT/CC mid to late March 2020.

XI. Conclusion

We demonstrated a novel attack on the BLE pairing in BT version 5.2. The attack utilizes a method we call Method Confusion to gain a MitM position between two paired Bluetooth devices. It abuses a critical design flaw that – upon our disclosure – was acknowledged by the Bluetooth SIG (see Section X).

The attack's applicability was verified by adopting a commonly used BLE driver into our framework BThack, capable of performing the MitM attack. Using this, we attacked multiple smartphones, a smartwatch, and a banking device. All attacks succeeded and we expect many other devices to be vulnerable. To verify the real-world impact of the attack, we also designed an end-to-end PoC that includes an advertisement jammer.

Furthermore, we evaluated if users are able to notice the attack by performing a user study with 40 participants. None of the participants noticed the attack and 92.5% faithfully completed the pairing leading to a MitM position.

We proposed multiple fixes to this problem. Some of them can be realized in a backward-compatible fashion. Others require a change to the specification.

Bluetooth is a major wireless communication standard and especially the number of BLE devices continues to increase rapidly. Many devices such as point-of-sale terminals and smartwatches require the protection promised by BLE. Our findings enable an attacker to steal and manipulate data and target otherwise protected APIs of such devices.

We hope that our work contributes to an increase in Bluetooth's security, so that device vendors can continue using it as a trusted building block for their products.

Availability

We intend to publish BThack and our PoC implementation of the Method Confusion Attack on https://github.com/maxdos64/BThack.

Acknowledgments

We would like to thank Matthias Ringwald from BlueKitchen GmbH for granting us permission to use BTstack. Further, we want to thank Jiska Classen and Constantin Runge for their constructive feedback on our research, and Yan Shoshitaishvili and Selcuk Uluagac for shepherding our work.

References

[1] *Bluetooth Core Specification 5.2*, Bluetooth Specification Contributors, December 2019.

[2] Bluetooth SIG, "Bluetooth market update 2019," https://3pl46c46ctx02p7rzdsvsg21-wpengine.netdna-ssl.com/wp-content/uploads/2018/04/2019-Bluetooth-Market-Update.pdf, 2019.

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

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

[5] K. Hypponen and K. Haataja, ""Nino" man-in-the-middle attack on Bluetooth secure simple pairing," in *2007 3rd IEEE/IFIP International Conference in Central Asia on Internet*, 2007, pp. 1–5.

[6] E. Biham and L. Neumann, "Breaking the Bluetooth pairing–fixed coordinate invalid curve attack," Technion–Israel Institute of Technology, Tech. Rep., 2018. [Online]. Available: http://www.cs.technion.ac.il/~biham/BT/bt-fixed-coordinate-invalid-curve-attack.pdf

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

[8] Bluekitchen GmbH, "Btstack." [Online]. Available: http://bluekitchen-gmbh.com/btstack/

[9] R. Cayre, V. Nicomette, G. Auriol, E. Alata, M. Kaâniche, and G. Marconato, "Mirage: Towards a Metasploit-like framework for IoT," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019. [Online]. Available: https://hal.laas.fr/hal-02346074

[10] R. Cayre and B. contributers, "Btlejack firmware," https://github.com/RCayre/btlejack-firmware, 2019.

[11] *nRF51 Series Reference Manual Version 3.0.1*, Nordic Semiconductor, December 2016.

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

[13] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, "Uncovering privacy leakage in BLE network traffic of wearable fitness trackers," in *17th International Workshop on Mobile Computing Systems and Applications*, 2016, p. 99–104. [Online]. Available: https://doi.org/10.1145/2873587.2873594

[14] B. Cyr, W. Horn, D. Miao, and M. Specter, "Security analysis of wearable fitness devices (fitbit)," *Massachusetts Institute of Technology*, vol. 1, 2014. [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2015/03/20082016/17-cyrbritt-webbhorn-specter-dmiao-hacking-fitbit.pdf

[15] T. Szigeti and C. Hattingh, "Quality of service design overview," *Cisco, San Jose, CA, Dec*, pp. 1–34, 2004.

[16] S. Schechter, "Common pitfalls in writing about security and privacy human subjects experiments, and how to avoid them," https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/commonpitfalls.pdf, 2013, Microsoft Research Technical Report.

[17] K. J. Kuipers and S. J. Hysom, "Chapter 7 – Common problems and solutions in experiments," in *Laboratory Experiments in the Social Sciences (Second Edition)*, M. Webster and J. Sell, Eds. Academic Press, 2014, pp. 145 – 177.

[18] *Guide to Bluetooth Security*, National Institute of Standards and Technology (NIST), 5 2017, revision 2.

[19] Bluetooth SIG, "Launch studio," last accessed: April 31, 2020. [Online]. Available: https://launchstudio.bluetooth.com/Listings/Search

[20] IDC, "Shipments of wearable devices reach 118.9 million units in the fourth quarter and 336.5 million for 2019, according to IDC," Mar. 2020. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS46122120

[21] Mordor Intelligence, "Smartwatch market – Growth, trends, and forecast (2020 - 2025)," Jan. 2020. [Online]. Available: https://www.mordorintelligence.com/industry-reports/global-smart-watches-market-industry

[22] Counterpoint Research, "Global smartwatch shipments grew 48% YoY in Q1 2019 with one in three being an Apple watch," May 2019. [Online]. Available: https://www.counterpointresearch.com/global-smartwatch-shipments-grew-48yoy-q1-2019-one-three-apple-watch/

[23] focus2move, "Global auto market. The ranking by model in the 2019," Jan. 2020. [Online]. Available: https://focus2move.com/world-best-selling-car-2019/

[24] E. Uzun, K. Karvonen, and N. Asokan, "Usability analysis of secure pairing methods," in *International Conference on Financial Cryptography and Data Security*. Springer, 2007, pp. 307–324.

[25] C. Fontaine, *E0 (Bluetooth)*. Boston, MA: Springer US, 2005, pp. 169–169.

[26] D. Whiting, R. Housley, and N. Ferguson, "Counter with CBC-MAC (CCM) RFC 3610," in *IETF, Network Working Group, Fremont*, 2003.

[27] D. Kügler, ""Man in the middle" attacks on Bluetooth," in *International Conference on Financial Cryptography*. Springer, 2003, pp. 149–161.

[28] M. Ryan, "Bluetooth: With low energy comes low security," in *7th USENIX Workshop on Offensive Technologies (WOOT)*, 2013.

[29] J. Barnickel, J. Wang, and U. Meyer, "Implementing an attack on Bluetooth 2.1+ secure simple pairing in passkey entry mode," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, 2012, pp. 17–24.

[30] *Erratum 10734: Pairing Updates*, Bluetooth Specification Contributors, July 2018.

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

[32] J. Ruge, J. Classen, F. Gringoli, and M. Hollick, "Frankenstein: Advanced wireless fuzzing to exploit new bluetooth escalation targets," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/ruge

[33] B. Seri and G. Vishnepolsky, "BlueBorne: The dangers of Bluetooth implementations: Unveiling zero day vulnerabilities and security flaws in modern Bluetooth stacks," Department of Computer Science, Michigan State University, East Lansing, Michigan, Tech. Rep., 2017. [Online]. Available: https://go.armis.com/blueborne-technical-paper

APPENDIX

A. Acronyms

BC Bluetooth Classic.
BLE Bluetooth Low Energy.
Bluetooth SIG Bluetooth Special Interest Group.

DH Diffie-Hellman.

EC Elliptic Curve.
ECDH Elliptic Curve Diffie-Hellman.

GAP Generic Access Profile.
GATT Generic Attribute Profile.

IOCap Input-Output Capabilities.
IPC Inter Process Communication.
IRK Identity Resolving Key.

JW Just Works.

LESC Low Energy Secure Connections.
LTK Long-Term Key.

MitM Man-in-the-Middle.

NC Numeric Comparison.
NoP Numeric on Passkey.

OOB Out of Band.

PE Passkey Entry.
PK Public Key.
PoC Proof of Concept.
PoN Passkey on Numeric.

RTT Round Trip Time.

SC Secure Connections.
SSP Secure Simple Pairing.

B. Survey & and Survey Results

In the following, the post-study survey and numeric answer data are summarized.

I. Demographics:

a) Age: on average 22.5

b) Gender: 35% female 65% male

c) Field of studies:

- Computer Science (60%)
- Math (10%)
- Physics (5%)
- Others (20%)
- No answer (5%)

d) Attained degree:

- Bachelor (60%)
- Master (35%)
- PhD (2.5%)
- Other (2.5%)

e) Number of semesters studying:

f) Which is the most interesting subtopic of your studies?:

II. Overall Technical Experience:

a) How many electronic devices do you use every day?:

- 1 (0%)
- 2-3 (37.5%)
- $>$ 3 (60%)
- I do not regularly use any electronic devices (0%)
- No answer (2.5%)

b) Do you regularly use a Laptop/Tablet/etc. at work/university?:

- Yes (100%)
- No (0%)

c) What is your main application for electronic devices?:

- Social media (37.5%)
- Email (32.5%)
- Office (Word/Excel/etc.) (27.5%)
- Programming (47.5%)
- Others (25%)

d) On a scale from 1 to 4, how experienced would your rate yourself with technical devices? (1 is no experience, 4 is expert):

- 1 (0%)
- 2 (12.5%)
- 3 (50%)
- 4 (32.5%)
- No answer (5%)

III. Wireless Experience:

a) Do you use any Bluetooth devices?:

- Keyboard/mouse (35%)
- Headset (65%)
- Smart Home (15%)
- Others (32.5%)

b) Are you used to the wireless pairing process?:

- Yes (85%)
- No (15%)

IV. Personal Opinion:

a) Apple does not support wired headphones anymore. Do you think this is a good idea?:

- Yes (12.5%)
- No (85%)
- No answer (2.5%)

b) Do you think Bluetooth pairing is simple to do, if not why?:

- Yes (80%)
- No (20%)

c) If no, why?:

d) Do you think wireless devices are an improvement over the common wired alternatives?:

- Yes (62.5%)
- No (37.5%)

e) On a scale from 1 to 4, how secure do you think wireless devices are? (1 is not secure, 4 is very secure):

- 1 (17.5%)
- 2 (57.5%)
- 3 (25%)
- 4 (0%)

f) Do you have any additional thoughts?:

C. Market Survey on spread of Bluetooth MitM protection

Samsung Gear Watches		
Galaxy Watch	https://youtu.be/8pSaUKQNN-E?t=67	NC
Galaxy Watch Active	https://youtu.be/_yGoPkB7jIA?t=201	NC
Galaxy Watch Active 2	https://youtu.be/6QS7R6Dpy80?t=126	NC
Galaxy Sport	https://youtu.be/nDVJbUEhTYk?t=220	NC
Galaxy Gear S3	https://youtu.be/Zd5IEFxw23Q?t=83	NC
Galaxy Gear S2	https://youtu.be/4Om5eh9OE38?t=522	NC
Galaxy Gear S	https://youtu.be/Lc8vv_T9Mas?t=69	NC
Galaxy Gear	https://youtu.be/oo8Yx9ZTfnw?t=248	NC
Fossil		
Fossile Smart Watch Gen 1-5/Sport (WearOS)	https://youtu.be/7qyp1Y7W1hw	NC
Garmin Smartwatches		
Garmin Vivoactive 4/4s	https://youtu.be/PvaFJVXb-R4?t=72	PE
Garmin Vivoactive 3/3s	https://youtu.be/T4DSrSkBkyc?t=58	PE
Garmin Vivomove	https://youtu.be/T4DSrSkBkyc?t=58	PE
Garmin Swim 2	https://youtu.be/bk7GRfLzHMs?t=64	PE
Garmin Forerunner 645	https://youtu.be/OxjXhTZgNXs?t=92	PE
Garmin Forerunner 45	https://youtu.be/3e2U_JD9F2M?t=88	PE
Garmin Forerunner 245	https://youtu.be/coBOyf3irY0?t=71	PE
Garmin Forerunner 945	https://youtu.be/diqRxRw_w_U?t=88	PE
Garmin Approach S40	https://youtu.be/lvq_eUrxVdE?t=46	PE
Garmin Fenix 5	https://youtu.be/DdRQCWzNW7s?t=48	PE
Garmin Fenix 6 / tactix	https://youtu.be/BNxyYGrAXuo?t=82	PE

Fig. 11. Market Survey Result: Smartwatches that rely on Bluetooth MitM protection

Toyota Corolla	https://www.toyota.com/content/entune/pdf/Bluetooth_setup.pdf	NC
Ford F-Series	https://owner.ford.com/support/how-tos/sync/sync/setup/how-to-connect-or-pair-my-phone-with-sync.html	PE
Toyota RAV4	https://www.toyota.com/content/entune/pdf/Bluetooth_setup.pdf	NC
Honda Civic	https://youtu.be/M1oQ9hNI630	NC
Honda CR-V	https://www.handsfreelink.com/Honda/en-US/US/PairYourPhone/Instructions?modelid=RW2H8KKNW&carrierid=107&phoneid=5473131	NC
Toyota Camry	https://www.toyota.com/content/entune/pdf/Bluetooth_setup.pdf	NC
Ram pick-up	https://www.mopar.com/en-us/technology/bluetooth-pairing.html	NC / PE
Toyota Hilux	https://www.toyota.com/content/entune/pdf/Bluetooth_setup.pdf	NC
Chevrolet Silverado	https://my.chevrolet.com/bluetooth	NC
Volkswagen Tiguan	https://youtu.be/cyvzrf7US2E?t=29	NC

Fig. 12. Market Survey Result: Car Entertainment Systems (Most popular models in 2019) that rely on Bluetooth MitM protection