Cryptographic Analysis of the Bluetooth Secure Connection Protocol Suite

Marc Fischlin Olga Sanina

Cryptoplexity, Technische Universität Darmstadt, Germany
www.cryptoplexity.de
{marc.fischlin,olga.sanina}@cryptoplexity.de

Abstract. We give a cryptographic analysis of the Bluetooth Secure Connections Protocol Suite. Bluetooth supports several subprotocols, such as Numeric Comparison, Passkey Entry, and Just Works, in order to match the devices' different input/output capabilities.

Previous analyses (e.g., Lindell, CT-RSA'09, or Troncoso and Hale, NDSS'21) often considered (and confirmed) the security of single subprotocols only. Recent practically verified attacks, however, such as the Method Confusion Attack (von Tschirschnitz et al., S&P'21) against Bluetooth's authentication and key secrecy property, often exploit the bad interplay of different subprotocols. Even worse, some of these attacks demonstrate that one cannot prove the Bluetooth protocol suite to be a secure authenticated key exchange protocol.

We therefore aim at the best we can hope for and show that the protocol still matches the common key secrecy requirements of a key exchange protocol if one assumes a trust-on-first-use (TOFU) relationship. This means that the adversary needs to mount an active attack during the initial connection, otherwise the subsequent reconnections remain secure.

Investigating the cryptographic strength of the Bluetooth protocol, we also look into the privacy mechanism of address randomization in Bluetooth (which is only available in the Low Energy version). We show that the cryptography indeed provides a decent level of address privacy, although this does not rule out identification of devices via other means, such as physical characteristics.

This is the full version of the paper appearing in the proceedings of the *27th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2021).* © IACR 2021,
DOI: 10.1007/978-3-030-92075-3_24.

Contents

1 Introduction **3**

1.1 Connecting Securely with Bluetooth . . . 3

1.2 A Short History of Attacks . . . 4

1.3 A Short History of Analyses . . . 6

1.4 Bluetooth as a TOFU Key Exchange Protocol . . . 6

1.5 Privacy . . . 7

1.6 Paper structure . . . 8

2 Bluetooth **8**

2.1 High-Level Protocol Flow . . . 9

2.2 Secure Simple Pairing . . . 10

2.3 Deriving the Encryption Key . . . 12

3 Security Model **14**

3.1 Attack Model . . . 14

3.2 Security Properties . . . 16

4 Security of Bluetooth **17**

4.1 Security Assumptions . . . 18

4.2 Match Security . . . 19

4.3 Key Secrecy . . . 20

5 Privacy in Bluetooth LE **22**

5.1 Details on Privacy Mechanisms in Bluetooth Low Energy . . . 23

5.2 Privacy Requirements . . . 25

5.3 Privacy Guarantees of BLE . . . 26

6 Conclusion **27**

A Other Subprotocols for Association Models in SSP **32**

A.1 Just Works . . . 32

A.2 Passkey Entry . . . 32

A.3 Out-of-Band . . . 33

B Acronyms, Glossary, and Variables **35**

B.1 Acronyms . . . 35

B.2 Glossary . . . 35

B.3 Variables . . . 38

1 Introduction

Bluetooth has become an omnipresent standard for short-range wireless communication. It is used in billions of products today: from powerful devices like computers and smartphones to more limited devices like headsets. The standard is maintained by the Bluetooth Special Interest Group and its latest specification of more than 3,000 pages describes version 5.4 [BT523][1].

The Bluetooth protocol comes in two major versions, the classical version (BR/EDR, for basic rate/enhanced data rate) and the low-energy version (BLE). The BR/EDR variant is usually used for connections with continuous data streams like headphones. In contrast, BLE is typically used when power consumption is a concern and data is only transferred periodically, e.g., for fitness trackers. The modes are not compatible but dual-mode devices are able to use both technologies.

1.1 Connecting Securely with Bluetooth

To transfer data between two Bluetooth devices securely and bidirectionally, they need to initially establish the link on a physical and logical level. If this has happened, then both devices establish a cryptographic key, called the link key in BR/EDR resp. long-term key in BLE. This key is used to derive an encryption key for communication following the link establishment as well as to authenticate devices and derive new encryption keys in later reconnections. In the latest version 5.4 of the standard [BT523], the strongest method to establish such a key is the Secure Connections (for BR/EDR) resp. LE Secure Connections (for BLE). Previous versions of (more or less secure) connection protocols are nowadays called legacy.

The main part of the Secure Connections protocol, so-called Secure Simple Pairing (SSP), has been added to BR/EDR already with version 2.1. With version 4.1, the SSP protocol has been upgraded to the Secure Connections protocol, using FIPS-approved cryptographic algorithms. BLE has been introduced in version 4.0 and has not inherited the protocol (and security) from classical Bluetooth. Only since version 4.2, BLE supports the Secure Connections pairing. The main difference between the Secure Connections methods in BR/EDR and BLE in terms of cryptographic operations is that Secure Connections for BR/EDR uses HMAC for message authentication and key derivation in the key-exchange part, whereas the LE version uses AES-CMAC. In the following high-level discussion, we thus lump both protocols together under the term Secure Connections.

The Secure Connections protocol itself is a protocol family, all members sharing an Elliptic-Curve Diffie–Hellman (ECDH) key exchange with key confirmation. Only the authentication stages differ, depending on the input/output capabilities of the connecting devices. For example, some devices may be able to display numbers, some only allow for a yes/no confirmation, and some may not support any interaction. Hence, there are four connection modes, also called association models:

Numeric Comparison: The devices display a short 6-digit number, which the user has to compare and confirm by pressing a button.

Passkey Entry: The user enters a 6-digit passkey on both devices (or, one device displays the passkey and the user enters the value into the other device).

Out-of-Band: Some device data is exchanged via an alternative channel, e.g., via a separate NFC connection between the two devices before the protocol execution.

Just Works: The devices connect without any further form of user involvement.

In the Bluetooth standard [BT523], the first three subprotocols (NumCom, PasskeyEntry, and OOB) are referred to as authenticated, whereas the JustWorks subprotocol is called unauthenticated.

[1] This work was conducted when the version 5.3 [BT521] of the Bluetooth Core Specification was the latest. However, the new version [BT523] does not affect the results of this work since the Standard has not changed the cryptographic component.

1.2 A Short History of Attacks

The Bluetooth protocol family has been repeatedly shown to be vulnerable to attacks. We only discuss here the most recent attacks, especially on the latest standards, which are also most relevant for our result. One goal of the adversary is to fool the authentication property of Bluetooth, ideally also allowing to learn the session key between the devices.

As pointed out by Zhang et al. [ZWD$^+$20], the PasskeyEntry subprotocol is susceptible to machine-in-the-middle attacks. The attack displayed in Figure 1 is based on the different input/output capabilities of devices. Here the user aims to connect a `KeyboardOnly` device (in this case, a keyboard) to a `DisplayOnly` device (in this case, a screen), allowing the attacker to connect its own keyboard to the user's screen, without being detected. This means that PasskeyEntry does not allow reliable authentication of devices.

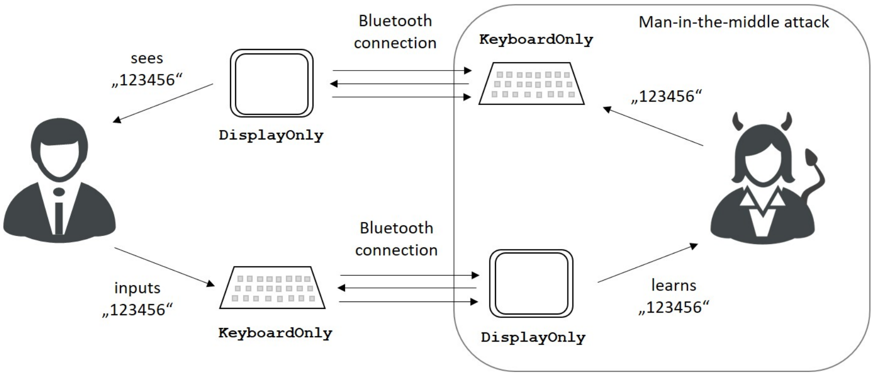

Figure 1: Machine-in-the-middle attack against authentication of devices, as presented in [ZWD$^+$20]. The attacker is able to connect its `KeyboardOnly` device to the user's `DisplayOnly` device by relaying the information between the devices. Note that, except for the passkey, the two Bluetooth connections are independent, enabling the adversary to know the long-term key in the upper connection.

Another problem with the PasskeyEntry protocol has been pointed out by Troncoso and Hale [TH21]. They discuss that the initiator- or responder-generated passkey protocol allows a machine-in-the-middle attacker to make two devices connect with the help of the user, but such that the two devices are cryptographically not partnered. For the user-generated PasskeyEntry case, they discuss a "role confusion" attack wherein both parties accept and believe to be the initiator of the connection.

Another active attack on the initial connection using the PasskeyEntry subprotocol has been presented by Claverie and Lopes-Esteve [CL21] and is called BlueMirror. In this machine-in-the-middle attack, the adversary reflects the data in the execution with the initiator to learn the passkey shared between two devices. Then the adversary uses the learned passkey in the connection with the responder, eventually making the responder believe to communicate with the original initiator. Still, the adversary holds the key in the execution with the responder.

The paper of von Tschirschnitz et al. [vTPFG21] introduced the Method Confusion Attack, which allows the adversary to place itself in the middle between two devices (Figure 2). The adversary establishes two connections with the devices by running the PasskeyEntry subprotocol in one session and the NumCom subprotocol in the other one. Since it can ask the user in the first connection (PasskeyEntry) to enter exactly the value used in the second connection (NumCom), the user(s) will confirm both connections. Eventually, the devices are thus considered to be connected, although they are each paired with the adversary. The attack is based on the fact that the passkeys both in NumCom and PasskeyEntry use the same length and alphabet, making it impossible for the user to distinguish the two subprotocols. Claverie et al. [CADLE23] later extended this attack to the Pairing Confusion Attack. There, the confusing included not only NumCom and PasskeyEntry but also legacy protocols, which employ 6-digit PIN that can be confused with the passkey.

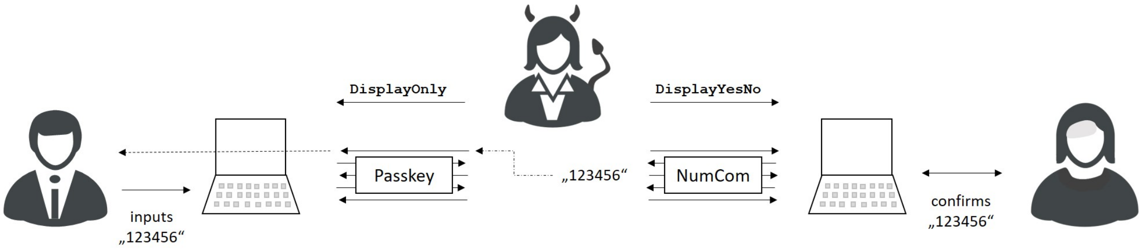

Figure 2: Machine-in-the-middle attack connecting two devices, as presented in [vTPFG21]. The attacker is able to make the passkey in the left connection match the numeric comparison value in the second one, prompting the user(s) to confirm pairing. The adversary eventually knows the keys of both connections, what allows intercepting and relaying of the communication.

The bad interplay of Bluetooth BR/EDR and Bluetooth LE has been exploited in the so-called BLUR attack [ATRP20]. If the devices establish a key in the BR/EDR or in the LE mode, then they can convert it to another key for the complementary mode (cross-transport key derivation), enabling a potential switch to the other architecture later. In [ATRP20], however, it has been demonstrated that an adversary can use this feature to overwrite the securely established key by an unauthenticated just-works key via the other connection mode.

The lack of authentication of the negotiation data enabled the “Key Negotiation of Bluetooth” (KNOB) attack discovered by Antonioli, Tippenhauer, and Rasmussen [ATR19, ATR20b], where the machine-in-the-middle adversary modifies the requested key length. It sets the entry to 1 byte for session keys in BR/EDR resp. 7 bytes for long-term and session keys in BLE, making the devices use weak keys that can be recovered by exhaustive search. This attack, as most of the previously mentioned ones, has also been demonstrated in practical scenarios.

With the Bluetooth Impersonation AttacksS (BIAS), Antonioli, Tippenhauer, and Rasmussen [ATR20a] have demonstrated that an adversary can enforce a reconnection for classical Bluetooth to any of two parties sharing a link key, without the adversary knowing the key. The attack exploits legacy authentication of BR/EDR, which does not enforce mutual authentication of partners and does not protect the switch the Central and Peripheral roles under the shared key. If this is the case, the adversary can connect to any of the two parties by asking one to switch roles and relaying the authentication information. For Secure Connections, the attack works if the devices support downgrade to legacy security because the request is not authenticated.

Bluetooth LE Spoofing Attack (BLESA) is a downgrade attack, described in [WNK+20]. The attack comes in two versions and has also been shown feasible in practice. One version of BLESA is on reactive authentication and lets the adversary make the partner device switch to an encryption-free data transfer in reconnections. The other version is against proactive authentication, exploiting that some implementations do not correctly close connections when being asked to downgrade the encryption level in reconnections. The former is a shortcoming in the design of the protocol, and the latter of the implementations.

Antonioli [Ant23] exploits the unilateral authentication in BR/EDR legacy reconnections and discovers “BLUetooth Forward and Future Secrecy” (BLUFFS) attacks. All the six found attacks root in reuse of the same nonce enforced by the adversary, leading to the same key being derived in several sessions. Combined with the KNOB attack [ATR19], the BLUFFS attack facilitates the exhaustive search for recovering the session key that is reused in several reconnection sessions. Employing BIAS [ATR20a] to downgrade to legacy reconnection lets the adversary to launch the same attack against reconnections in Secure Connections mode.

We conclude this section by noting that, so far, the OOB subprotocol has not displayed major vulnerabilities. But this may have to do with the fact that any such attack, likewise any positive security result, would need to make additional assumptions about the extra communication channel. Furthermore, this mode seems to be also much less prominent than the other modes, as it requires additional communication means like NFC or optical components to scan QR codes.

1.3 A Short History of Analyses

Despite the attacks above, the literature also reveals a number of affirmative security results. The mismatch to the above attacks often relies on the fact that the attacks exploit vulnerabilities between different pairing subprotocols (e.g., associating PasskeyEntry and NumCom in the Method Confusion Attack [vTPFG21]), or between the BR/EDR and Low Energy cross-transport mode (like the BLUR attack [ATRP20]), or forcing the devices to switch to weak legacy subprotocols (like the BIAS attack [ATR20a]). In contrast, most cryptographic analysis focus on a single subprotocol only.

In [Lin09], Lindell studies Bluetooth's Numeric Comparison protocol as a key exchange protocol (in Bluetooth specification v2.1 but the cryptographic differences to the current version are minor). He shows that the NumCom protocol—as a standalone protocol—is a secure (comparison-based) key exchange protocol under the DDH assumption and further modest assumptions about the underlying primitives. Noteworthy, the model somehow assumes that user confirmation of the comparison value also authenticates the Bluetooth addresses, although these data are transmitted unprotected over the network and are not displayed to the user.

Sun and Sun [SS19] extended the result of Lindell to BR/EDR in version v5.0, for NumCom and OOB as standalone protocols. They reach the same conclusions in terms of security as [Lin09] for these protocols. Yet, their security model is more restrictive (e.g., the adversary is not allowed to communicate with parties after the test query).

We have already mentioned the analysis of Troncoso and Hale [TH21] in the attack section above. Noting the insecurities in the PasskeyEntry subprotocol, they give a security proof for two modified versions of PasskeyEntry, also as a standalone protocol. The first modification, secure hash modification, includes more data in the hash computation. The other modification, the dual passkey entry, presumes that both devices allow entering and displaying a passkey. Both versions are shown to be secure under the DDH assumption, reasonable assumptions about the other cryptographic primitives, and a single-query version of the PRF-ODH assumption [JKSS12].

1.4 Bluetooth as a TOFU Key Exchange Protocol

The starting point of our approach originates from the observation that known attacks show Bluetooth, as a full protocol suite, does not provide authentication of keys. There is no chance to show security in the common sense of authenticated key exchange. This either leaves us with analyzing a modified protocol (as in [TH21])—and strictly speaking thus not giving any security guarantees for Bluetooth—or to switch to the best security claim "we can hope for". We decided for the latter.

We analyze Bluetooth as a *trust-on-first-use* (TOFU) authenticated key exchange protocol according to a BR-like security model. This means we assume that the adversary is passive in the initial connection and can only mount active attacks on devices that have been bonded (i.e., derived and store a respective link or long-term key) before. Of course, the adversary may on top bond arbitrarily with all the devices, but such interactions are, by definition, not protected since no trust relationship has been established. Besides capturing all possible pairing subprotocols simultaneously, we note that this also extends previous analyses by the reconnection step.

While the guarantees as a TOFU protocol appear to be quite weak, superficially viewed, it gives quite assuring guarantee for “minimalistic” modes of operations. That is, suppose that one significantly reduces attack vectors by turning off the compatibility features: specifically, no legacy protocols but only Secure Connections, sufficient key lengths, no cross-transport key derivation between BR/EDR and BLE. Then the TOFU result says that successful attacks against session keys can only be mounted if the adversary is present when the devices are initially connecting.

Our analysis assumes to be “close to the standard”. For instance, the security analyses in [Lin09, SS19, TH21] assume that the parties use a fresh Diffie–Hellman share in each execution. The Bluetooth v5.4 standard, however, allows the Diffie–Hellman key to be reused in several executions [BT523, Vol 2, Part H, Section 5.1]:

> “...a device should change its private key after every pairing (successful or failed). Otherwise, it should change its private key whenever S + 3F > 8, where S is the number of successful pairings and F the number of failed attempts since the key was last changed.”

Note that this explicitly refers to the Elliptic-Curve Diffie–Hellman (ECDH) public-private key pair generated in the first phase of the SSP protocol [BT523, Vol 2, Part H, Section 7.1]. In particular, in [Lin09] Lindell identifies partnered sessions via the public Diffie–Hellman shares of the partners. Since two devices may reuse their shares multiple times but choose different nonces in these initial connections (and thus derive different keys), strictly speaking, Lindell’s result cannot even guarantee basic correctness properties for the real Bluetooth protocol.

Another deviation from the standard is that the analyses in [Lin09, SS19] assume the entire Diffie–Hellman curve point enters the protocol computations, whereas the standard only uses the x-coordinate of the elliptic-curve point. Being aware of the possibility to enable attacks by this mapping, such as the fixed coordinate invalid curve attack [BN19], Troncoso and Hale [TH21] correctly use the x-coordinate in their protocol description.

1.5 Privacy

Bluetooth Low Energy supports a privacy mechanism that should help to disguise the device’s Bluetooth address during discovery. Essentially, instead of sending the physical MAC address, BLE permits to send a randomized address, either randomly generated only once during fabrication or each time when powering up the device, or refreshed in short time intervals. The latter type are called non-resolvable private random addresses. The protocol also has an advanced feature called resolvable private random addresses where a previously bonded device can recognize the pseudorandom address and link it to a physical MAC address.

In contrast, classical Bluetooth does not support address randomization or any other privacy mechanism. According to [CGP^{+}20], it was believed that tracking devices is hard, due to the larger number of communication channels and highly frequent channel hopping. This belief has recently been shown to be false in [CGP^{+}20]. The authors demonstrate that one can track devices even over large distances. Since the (de-)anonymization of BR/EDR devices escapes a cryptographic treatment, we focus here on the privacy mechanisms in BLE.

We are interested in the address-randomization technique and privacy on a protocol (i.e. transcript) level. Sun et al. [SSY19] provide an analysis of the BLE protocol, pointing out correctly that reusing the Diffie–Hellman key share in Secure Connections allows linking executions of different devices. They also provide a cryptographic analysis of privacy guarantees on the protocol layer, under the assumption that a fresh Diffie–Hellman value is used in each session. This analysis, however, neglects that other connection data (such as transmitting the Bluetooth address) may also allow the adversary to link executions of the same party. In particular, they do not consider BLE’s address-resolution technique but focus on the pairing stage only.

Besides inspecting the payload, an attacker may be able to distinguish devices according to their physical characteristics. This question recently gained attention in light of contact tracing via Bluetooth. For instance, Ludant et al. [LVHNN21] showed that dual-mode devices supporting Bluetooth BR/EDR (sending the plain address `BD_ADDR`) and BLE (potentially using randomized addresses) can be cross-linked by their channel characteristics for each of the two modes with high accuracy. This implies that the privacy mechanism of BLE effectively becomes void because of the lack of privacy for classical Bluetooth. Countermeasures may be to temporarily turn off either of the two unused transport or to reduce the transmission power in order to limit the attack radius.

Jouans et al. [JVAF21] demonstrated that the address randomization technique itself can actually be used against privacy: The frequency with which devices change their addresses can be used to differentiate them. Celosia and Cunche [CC19a] discuss that between 0.06% and 1.7% of devices using address randomization nonetheless transmit linkable cleartext names of devices. Another often encountered entry in the advertisement data is the Universally Unique Identifier (UUID) field to identify services and characteristics of the device. These 16, 32 or 128-bit values are usually available in the generic attribute profile (GATT) of the device and can be transmitted as part of the advertisement. Following similar attacks on Wi-Fi [VMC^{+}16] and BLE [BLS19], it has been pointed out in [CC19b] that the UUIDs can be used to fingerprint devices and overcome privacy techniques with address randomization.

Our analysis does not aim to protect against attacks based on the physical characteristics, but only to ensure that the cryptographic and privacy mechanisms do not support privacy breaches. The other distinctive characteristics must be taken care of by different means, e.g., using identical address randomization intervals on each device, or switching off cleartext name advertisements. We show that if the Diffie–Hellman values are chosen afresh in each execution, then the cryptographic technique of address randomization indeed provides the decent level of privacy.

1.6 Paper structure

The rest of the paper is organized as follows. We proceed with describing the Bluetooth protocol in Section 2. Starting with the high-level overview of the protocol, we give more details on its part called Secure Simple Pairing. While we focus on the Numeric Comparison subprotocol in this section, we leave the other subprotocols for Appendix A. The security model is given in Section 3 wherein we present the security notions for TOFU-authenticated key exchange. We apply the model to the Bluetooth protocol suite in Section 4. We first present the underlying cryptographic assumptions and then show that, under these assumptions, Bluetooth achieves Match Security and TOFU Key Secrecy. In Section 5, we investigate the privacy mechanism in BLE. We first give the details on the data exchanged during the protocol flow, which can be used to link the device. Then we present the definition of the outsider privacy and show that Bluetooth achieves it with a decent level. To help the reader throughout the paper, we give the list of acronyms, present the glossary, and describe the variables from the Bluetooth protocol in Appendix B.

2 Bluetooth

We start by giving an overview over the Bluetooth protocol suite along the standard [BT523]. The Bluetooth protocol comes in several versions with minor differences. The two basic connection protocols correspond to the transport types, which are Bluetooth *Basic Rate/Enhances Data Rate* (BR/EDR), also called Bluetooth Classic, and *Low Energy* (BLE). From a high-level cryptographic view point, the only differences are that in the pairing step, BR/EDR uses HMAC-SHA256 to compute the link key, whereas BLE uses AES-CMAC for the respective computation of the long-term key. In the reconnection step, however, the two transports diverge in the way they derive the session keys. Finally, BLE supports a privacy mechanism to hide the devices' addresses. We discuss the latter in Section 5.

We note that both transport protocols, BR/EDR and BLE, gradually converge to one protocol, while previous versions (“legacy versions”) had major differences. For instance, earlier versions of BLE did not use elliptic-curve DH mechanisms. Both transports are incompatible from a technological viewpoint, e.g., they use a different number of communication channels. Dual-mode devices, which support both technologies simultaneously, are becoming more and more ubiquitous.

2.1 High-Level Protocol Flow

The flow of two devices connecting in both versions, BR/EDR and BLE, is identical from an abstract viewpoint but differs in the technological aspects. We give a description of the relevant protocol parts in Figure 3. Initially both devices need to connect physically and logically. This is done in an inquiry or discovery phase and involves the devices exchange their Bluetooth addresses BD_ADDR. The address itself is a 48-bit value. To distinguish cleartext addresses from randomized ones in BLE, the devices uses the TxAdd and RxAdd (transmission/reception) flags, which we discuss in more detail when investigating the privacy feature in Section 5.

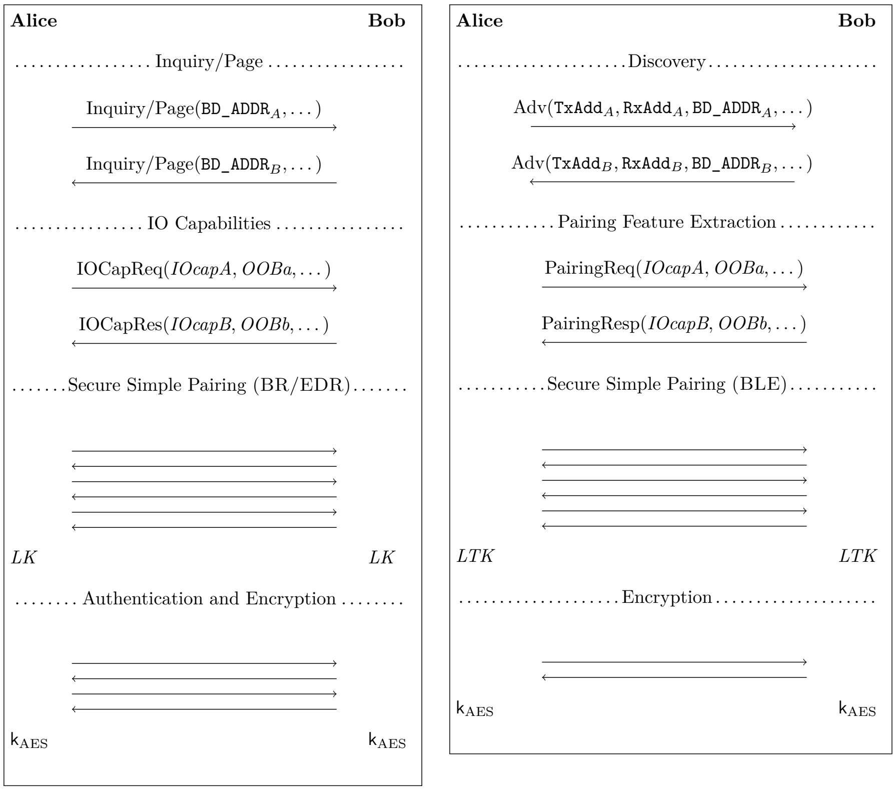
Alice
Bob
Inquiry/Page
Inquiry/Page(BD_ADDR$_A$, . . .)
Inquiry/Page(BD_ADDR$_B$, . . .)
IO Capabilities
IOCapReq($IOcapA$, $OOBa$, . . .)
IOCapRes($IOcapB$, $OOBb$, . . .)
Secure Simple Pairing (BR/EDR)
LK
LK
Authentication and Encryption
k_{AES}
k_{AES}
Alice
Bob
Discovery
Adv(TxAdd$_A$, RxAdd$_A$, BD_ADDR$_A$, . . .)
Adv(TxAdd$_B$, RxAdd$_B$, BD_ADDR$_B$, . . .)
Pairing Feature Extraction
PairingReq($IOcapA$, $OOBa$, . . .)
PairingResp($IOcapB$, $OOBb$, . . .)
Secure Simple Pairing (BLE)
LTK
LTK
Encryption
k_{AES}
k_{AES}

Figure 3: Bluetooth Protocol Flow (left: BR/EDR, right: BLE).

Then the devices connect on the link layer and can start exchanging device-specific information, especially the input/output capabilities. We note that BR/EDR and BLE use different commands for this, but we neglect these details here. In this step, the devices also exchange information about the strength of the connection (e.g., the SC flag in the feature vector in BLE to request Secure Connections, see Section 5.1). We assume that both devices only allow the strongest version called Secure Connections.

Based on the available IO capabilities, the devices decide on the subprotocol for Secure Simple Pairing (SSP) protocol, also called the association model. These IO capabilities determine how the device is able to interact with users. It can be either of the following five options: `DisplayOnly` (no input capability, numeric output), `DisplayYesNo` (yes/no input and numeric output), `KeyboardOnly` (keyboard input, no output), `NoInputNoOutput` (neither output nor input capabilities, or yes/no input and no output). The BLE protocol also supports `KeyboardDisplay` (keyboard input, numeric output). We note that one sometimes considers the exchange of the IO capabilities to be part of the SSP protocol, but this distinction is irrelevant for us here. The combination of the capabilities of the two devices determines the SSP subprotocol according to Table 1 (for Secure Connections only).

Table 1: Mapping of IO capabilities to association models. The last column and row `KeyboardDisplay` is only available in BLE.

	Initiator				
Responder	`DisplayOnly`	`DisplayYesNo`	`KeyboardOnly`	`NoInputNoOutput`	`KeyboardDisplay`
`DisplayOnly`	JustWorks	JustWorks	PasskeyEntry	JustWorks	PasskeyEntry
`DisplayYesNo`	JustWorks	NumCom	PasskeyEntry	JustWorks	NumCom
`KeyboardOnly`	PasskeyEntry	PasskeyEntry	PasskeyEntry	JustWorks	PasskeyEntry
`NoInputNoOutput`	JustWorks	JustWorks	JustWorks	JustWorks	JustWorks
`KeyboardDisplay`	PasskeyEntry	NumCom	PasskeyEntry	JustWorks	NumCom

We note that either device may set the out-of-band (*OOB*) flag as part of the features. In BR/EDR, this is part of the *IOcap* structure, whereas in BLE, this is a flag in the pairing features. If either device sets the *OOB* flag, then the parties use the OOB association model. We note that only one of the two devices may set this flag, in which case, only this device transmits out-of-band information. The data in the OOB association model contains the Bluetooth address of a device, commitments of the public keys, and random values that are used in further execution.

Next, the two devices execute the SSP protocol in the corresponding association model to establish a shared key. The steps are very similar and only differ in some cryptographic operations. We discuss the details in Section 2.2. For BR/EDR, the derived key is called a link key *LK*; for BLE, it is called a long-term key *LTK*. We note that both versions allow converting the key for future use in the other type of connection (cross-transport key derivation), which we do not consider here. This concludes the initial-connection procedure.

The final step is to derive the key for the authenticated encryption scheme. We note that this is also the protocol that is executed for reconnections, i.e., if the devices have bonded and created a shared key, and in this case, skip the SSP step. Here, the two protocols differ, as BR/EDR involves an additional authentication step. We discuss this part in more detail in Section 2.3.

2.2 Secure Simple Pairing

We next describe Secure Simple Pairing and its four variants: JustWorks, OOB, NumCom, and PasskeyEntry. At this point, the parties have already exchanged their 48-bit addresses A and B, their IO-capability values (leading to the agreement on the variant), and the elliptic curve to be used. In BR/EDR, if both devices agree on the Secure Connections mode, then the devices use the P-256 elliptic curve, else the P-192 curve. Both curves are FIPS-approved and defined in the Bluetooth standard. In BLE, only P-256 elliptic curve is used (in Secure Connections mode). For the elliptic-curve operations, we use the "simple" multiplicative presentation. That is, we write g^a for the a-fold application of the group operation to the generator g specified in the standard, without giving any further reference to the group. When processing elliptic-curve points in HMAC or CMAC in Authentication stage 1 of the SSP protocol, the standard uses the x-coordinate, i.e., we write $[g^a]_x$ for the x-coordinate of g^a. This x-coordinate is a 256-bit value for Secure Connections.

Similarly, when computing the Diffie–Hellman key via the functions P192 resp. P256 in the specification, only the x-coordinate of the resulting Diffie–Hellman value is used as a shared raw key [BT523, Vol 2, Part

H, Section 7.7.4]. We have to account for this by defining partnered sessions later via the x-coordinates of the exchanged values g^a and g^b only. Note that *for valid curve points*[2] these x-coordinates $[g^a]_x$ and $[g^b]_x$ fully determine the x-coordinate $[g^{ab}]_x$ of the Diffie–Hellman shared key g^{ab}. The reason is that the x-coordinate $[g^a]_x$ of an elliptic-curve point g^a determines the y-coordinate up to the sign, and that this sign tells us if the point is g^a or its inverse g^{-a}. Hence, given only $[g^a]_x$ and $[g^b]_x$, the potentially correct values are among $g^{\pm a}$ and $g^{\pm b}$. Since the actual Diffie–Hellman value is then either g^{ab} or g^{-ab} (where these points again share the same x-coordinate), any signs for $g^{\pm a}$ and $g^{\pm b}$ yield the same x-coordinate of the key share.

To capture both versions of the SSP protocol for BR/EDR and BLE simultaneously, we use abstract cryptographic procedures for computing the commitment value (Com), hashing (Hash), MAC-key computation (MACKey), MAC computation (MAC), and link key/long-term key computation (KDF). Roughly, for BR/EDR, these algorithms are initialized by HMAC-SHA256 (except for Hash, which uses SHA256 directly), and for BLE, one uses AES-CMAC. The different implementations of the primitives for BR/EDR and BLE are displayed in Table 2. We note that for the MAC-key computation in BR/EDR, the Diffie–Hellman value (here denoted W) can be used directly as a key in the HMAC computation MAC, since HMAC is able to process large keys. For AES-CMAC in BLE, however, the MAC key is computed via $\mathsf{CMAC}(\mathrm{Salt}, W)$ for a constant Salt and then used as a 128-bit key in the AES-CMAC computation of MAC.

Table 2: Cryptographic operations of BR/EDR and BLE in SSP. Note that $T = \mathsf{CMAC}(\mathrm{Salt}, W)$ for a fixed constant Salt in the standard; $\mathrm{kID}_{\mathrm{BR/EDR}} =$ `0x62746C6B` is a 4-octet representing the ASCII string 'btlk'; $\mathrm{kID}_{\mathrm{BLE}} =$ `0x62746C65` is a 4-octet representing the ASCII string 'btle'; for an address A in BLE the address A' is A extended by another octet `0x01` for a random address and `0x00` for a public address; the notation $/2^{128}$ for BR/EDR means that one takes the leftmost 128 bits of the SHA256 output.

Function	BR/EDR	BLE
$\mathsf{Com}(U, V, X, Y)$	$\mathsf{HMAC}(X, U \mid V \mid Y)/2^{128}$	$\mathsf{CMAC}(X, U \mid V \mid Y)$
$\mathsf{Hash}(U, V, X, Y)$	$\mathsf{SHA}(U \mid V \mid X \mid Y)$	$\mathsf{CMAC}(X, U \mid V \mid Y)$
$\mathsf{MACKey}(W, N1, N2, A1, A2)$	W	$\mathsf{CMAC}(T, \mathtt{0x00} \mid \mathrm{kID}_{\mathrm{BLE}} \mid N1 \mid N2 \mid A1' \mid A2' \mid \mathtt{0x0100})$
$\mathsf{MAC}(W, N1, N2, R, I, A1, A2)$	$\mathsf{HMAC}(W, N1 \mid N2 \mid R \mid I \mid A1 \mid A2)/2^{128}$	$\mathsf{CMAC}(W, N1 \mid N2 \mid I \mid A1' \mid A2')$
$\mathsf{KDF}(W, N1, N2, A1, A2)$	$\mathsf{HMAC}(W, N1 \mid N2 \mid \mathrm{kID}_{\mathrm{BR/EDR}} \mid A1 \mid A2)/2^{128}$	$\mathsf{CMAC}(T, \mathtt{0x01} \mid \mathrm{kID}_{\mathrm{BLE}} \mid N1 \mid N2 \mid A1' \mid A2' \mid \mathtt{0x0100})$

Figure 4 shows the Numeric Comparison protocol with the abstract operations. The NumCom protocol starts with the devices exchanging the Diffie–Hellman values, where the secret values must be picked from a restricted interval instead of $\mathbb{Z}_q$. Namely, the secret keys shall be picked between 1 and $q/2$. This is followed by Authentication stage 1 wherein the parties exchange random nonces and involve the user to confirm a 6-digit number Va resp. Vb. For this, a device truncates the hash value over the (x-coordinates of the) public-key parts and the nonces to 32 bits and then converts this to a decimal number. The last 6 digits correspond to the check values. It is followed by Authentication stage 2 in which the parties confirm the shared Diffie–Hellman key. Finally, both parties compute the link key (in BR/EDR) resp. the long-term key (in BLE).

We give more details on the other association models in Appendix A. These protocols only differ in the Authentication stage 1 of the SSP framework, which turns out to be irrelevant for our TOFU-security analysis. We merely remark that all association models, among others, exchange random nonces Na and Nb. We note that, technically, BLE computes the MAC key and long-term key in one step. We have moved the computation of the long-term key to the end of the protocol in order to comply with the BR/EDR step for computing the link key there.

[2] Since each party checks that the received point is on the curve, one can in particular exclude the invalid curve attack [BN19].

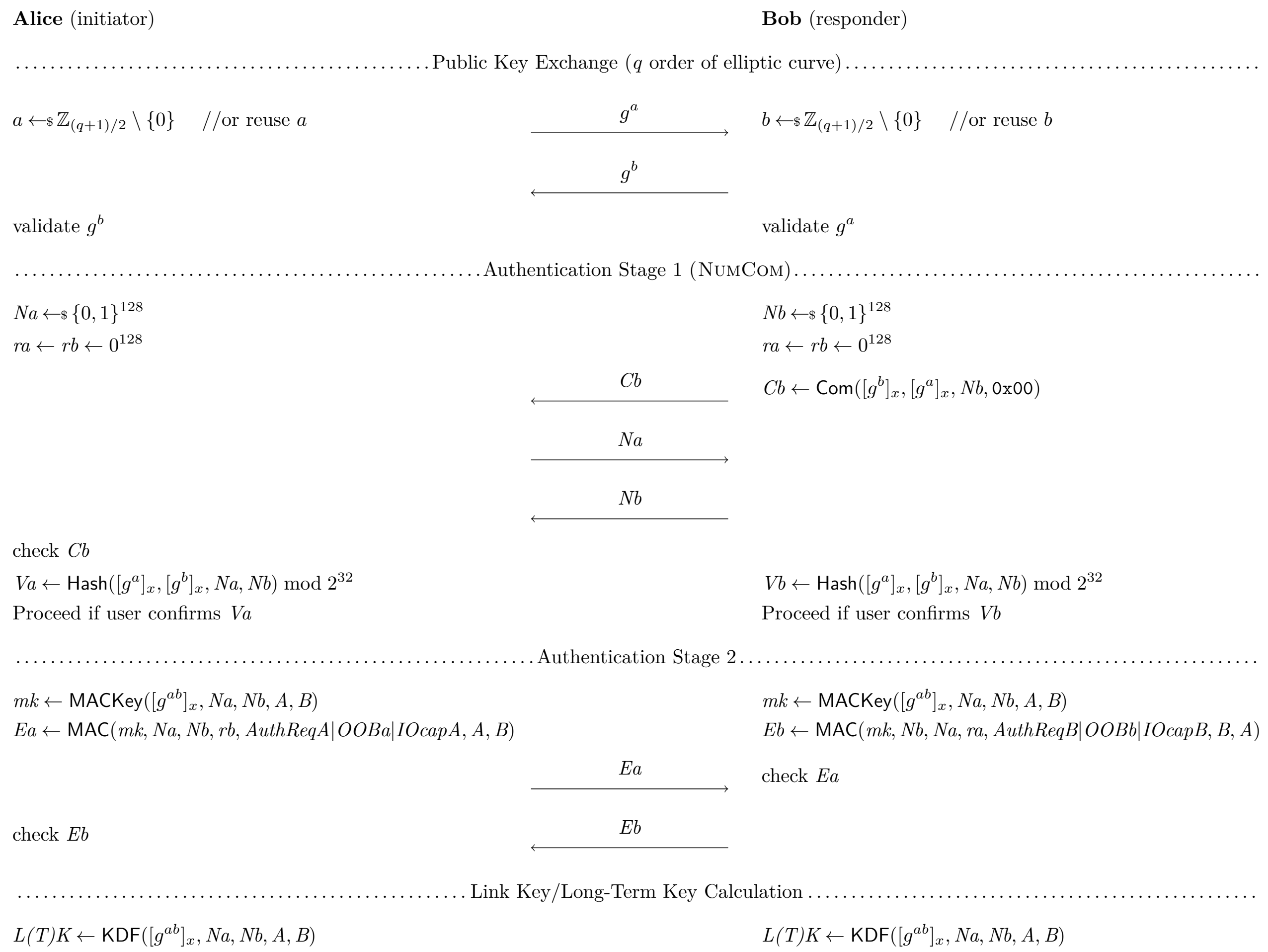
Alice (initiator)
Bob (responder)
Public Key Exchange (q order of elliptic curve)
$a \leftarrow_{\$} \mathbb{Z}_{(q+1)/2} \setminus \{0\}$ //or reuse a
$\xrightarrow{g^a}$
$b \leftarrow_{\$} \mathbb{Z}_{(q+1)/2} \setminus \{0\}$ //or reuse b
$\xleftarrow{g^b}$
validate g^b
validate g^a
Authentication Stage 1 (NumCom)
$Na \leftarrow_{\$} \{0,1\}^{128}$
$ra \leftarrow rb \leftarrow 0^{128}$
$Nb \leftarrow_{\$} \{0,1\}^{128}$
$ra \leftarrow rb \leftarrow 0^{128}$
$\xleftarrow{Cb}$
$Cb \leftarrow \mathsf{Com}([g^b]_x, [g^a]_x, Nb, \mathtt{0x00})$
$\xrightarrow{Na}$
$\xleftarrow{Nb}$
check Cb
$Va \leftarrow \mathsf{Hash}([g^a]_x, [g^b]_x, Na, Nb) \bmod 2^{32}$
Proceed if user confirms Va
$Vb \leftarrow \mathsf{Hash}([g^a]_x, [g^b]_x, Na, Nb) \bmod 2^{32}$
Proceed if user confirms Vb
Authentication Stage 2
$mk \leftarrow \mathsf{MACKey}([g^{ab}]_x, Na, Nb, A, B)$
$Ea \leftarrow \mathsf{MAC}(mk, Na, Nb, rb, AuthReqA|OOBa|IOcapA, A, B)$
$mk \leftarrow \mathsf{MACKey}([g^{ab}]_x, Na, Nb, A, B)$
$Eb \leftarrow \mathsf{MAC}(mk, Nb, Na, ra, AuthReqB|OOBb|IOcapB, B, A)$
$\xrightarrow{Ea}$
check Ea
check Eb
$\xleftarrow{Eb}$
Link Key/Long-Term Key Calculation
$L(T)K \leftarrow \mathsf{KDF}([g^{ab}]_x, Na, Nb, A, B)$
$L(T)K \leftarrow \mathsf{KDF}([g^{ab}]_x, Na, Nb, A, B)$

Figure 4: Bluetooth Secure Simple Pairing in mode Numeric Comparison. The session identifier, here and in all other association models, is given by $\mathsf{sid} = ([g^a]_x, [g^b]_x, A, B, Na, Nb)$.

2.3 Deriving the Encryption Key

The encryption key is derived differently in Bluetooth Classic and Low Energy. In the classical setting, it corresponds to a mutual challenge-response authentication protocol for the link key, which also enters the derivation of the session key (usually called AES encryption key in the Bluetooth context, although it serves as input to the AES-CCM authenticated-encryption scheme). That is, the parties exchange the 128-bit random values (AU_RAND), and each party computes the so-called 32-bit signed response (SRES) for authentication. In BLE instead, one simply derives the session key from (concatenated) 64-bit nonces, called session key diversifier (SKD), without further authentication.

BLE also uses AES-CCM for authenticated encryption of data. Both procedures produce some initial-nonce offset of 64 bits for the encryption process, denoted as ACO in BR/EDR and IV in BLE. In the latter case, the IV is given by the concatenation of the two random 32-bit values IV_C, IV_P, chosen by either party. From a security viewpoint, while ACO is not transmitted in clear, the IV in BLE is known by the adversary.

The steps for BR/EDR are described in Table 3 and Figure 5, and for BLE in Figure 6. We use the common notation of *Central* and *Peripheral* since the devices may change roles for reconnections. We note that in BLE, the key-derivation step and the data (SKD_C,IV_C resp. SKD_P,IV_P) are transmitted as part of an encryption-request/response message. In BR/EDR, the sequence must be preceded by an encryption_mode request and response. It is noteworthy that, in contrast to BLE where the key length is negotiated as part of the pairing-feature extraction, the BR/EDR protocol may negotiate the key length only here as well. We assume in the following that only the maximal key size is enforced by the devices, in order to prevent KNOB-like attacks [ATR19, ATR20b].

Table 3: Secure Authentication and Computation of Encryption Key in BR/EDR Secure Connections. HMAC is HMAC with SHA256; kID_{Dev} = `0x6274646B` is a 4-octet representing the ASCII string 'btdk' (Bluetooth Device Key); kID_{AES} = `0x6274616B` is a 4-octet representing the ASCII string 'btak' (Bluetooth AES Key); $SRES\_C, SRES\_P$ are 32 bits each, and ACO (Authentication Ciphering Offset) is 64 bits; the notation $/2^{128}$ means that one takes the leftmost 128 bits of the SHA256 output.

Value	Function
Device Key	$\text{dk} \leftarrow \mathsf{HMAC}(LK, \text{kID}_{\text{Dev}}\vert\texttt{BD\_ADDR}_A\vert\texttt{BD\_ADDR}_B)/2^{128}$
Confirmation	$SRES\_C\vert SRES\_P\vert\text{ACO} \leftarrow \mathsf{HMAC}(\text{dk}, AU\_RAND\_C\vert AU\_RAND\_P)/2^{128}$
AES Key	$\mathsf{k}_{\text{AES}} \leftarrow \mathsf{HMAC}(LK, \text{kID}_{\text{AES}}\vert\texttt{BD\_ADDR}_A\vert\texttt{BD\_ADDR}_B\vert\text{ACO})/2^{128}$

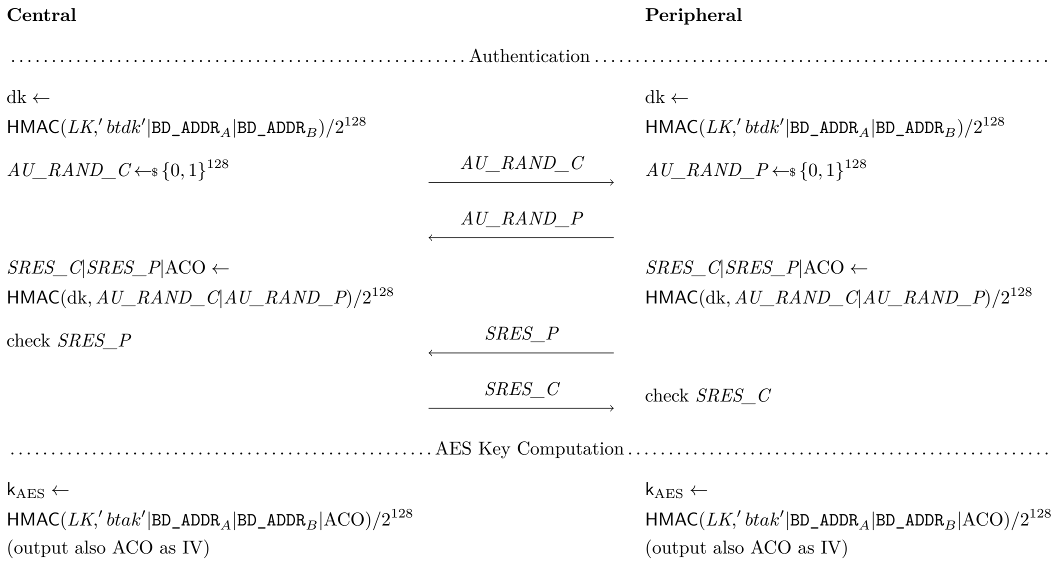
Central
Peripheral
Authentication
$\text{dk} \leftarrow$
$\mathsf{HMAC}(LK, 'btdk'\vert\texttt{BD\_ADDR}_A\vert\texttt{BD\_ADDR}_B)/2^{128}$
$\text{dk} \leftarrow$
$\mathsf{HMAC}(LK, 'btdk'\vert\texttt{BD\_ADDR}_A\vert\texttt{BD\_ADDR}_B)/2^{128}$
$AU\_RAND\_C \leftarrow_\$ \{0,1\}^{128}$
AU_RAND_C
$AU\_RAND\_P \leftarrow_\$ \{0,1\}^{128}$
AU_RAND_P
$SRES\_C\vert SRES\_P\vert\text{ACO} \leftarrow$
$\mathsf{HMAC}(\text{dk}, AU\_RAND\_C\vert AU\_RAND\_P)/2^{128}$
$SRES\_C\vert SRES\_P\vert\text{ACO} \leftarrow$
$\mathsf{HMAC}(\text{dk}, AU\_RAND\_C\vert AU\_RAND\_P)/2^{128}$
check $SRES\_P$
$SRES\_P$
$SRES\_C$
check $SRES\_C$
AES Key Computation
$\mathsf{k}_{\text{AES}} \leftarrow$
$\mathsf{HMAC}(LK, 'btak'\vert\texttt{BD\_ADDR}_A\vert\texttt{BD\_ADDR}_B\vert\text{ACO})/2^{128}$
(output also ACO as IV)
$\mathsf{k}_{\text{AES}} \leftarrow$
$\mathsf{HMAC}(LK, 'btak'\vert\texttt{BD\_ADDR}_A\vert\texttt{BD\_ADDR}_B\vert\text{ACO})/2^{128}$
(output also ACO as IV)

Figure 5: Bluetooth BR/EDR Secure Authentication and Encryption Key Derivation. The session identifier for this subprotocol is given by $\mathsf{sid} = (AU\_RAND\_C, AU\_RAND\_P, A, B)$.

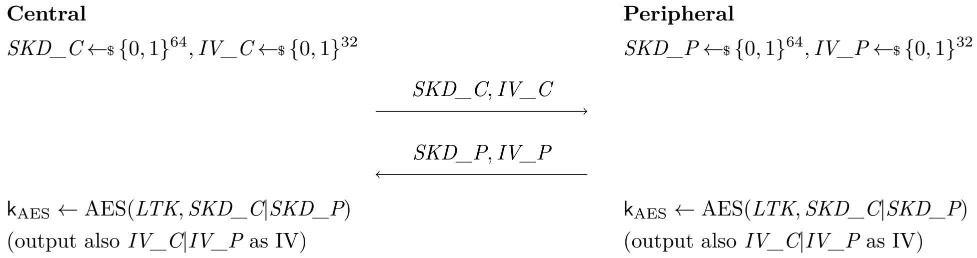
Central
$SKD\_C \leftarrow_\$ \{0,1\}^{64}, IV\_C \leftarrow_\$ \{0,1\}^{32}$
Peripheral
$SKD\_P \leftarrow_\$ \{0,1\}^{64}, IV\_P \leftarrow_\$ \{0,1\}^{32}$
SKD_C, IV_C
SKD_P, IV_P
$\mathsf{k}_{\text{AES}} \leftarrow \text{AES}(LTK, SKD\_C\vert SKD\_P)$
(output also $IV\_C\vert IV\_P$ as IV)
$\mathsf{k}_{\text{AES}} \leftarrow \text{AES}(LTK, SKD\_C\vert SKD\_P)$
(output also $IV\_C\vert IV\_P$ as IV)

Figure 6: Bluetooth BLE Encryption Key Derivation. The session identifier is given as $\mathsf{sid} = (SKD\_C, SKD\_P)$.

3 Security Model

In this section, we define our security model for TOFU-key-exchange protocols. Given the history of successful attacks against Bluetooth, especially against authentication, we aim at very basic security of key secrecy. Since Bluetooth does not achieve forward secrecy—if the link key resp. long-term key is available then all previous connections become insecure—we do not incorporate this feature into our model. We also note that it is convenient to model the initial-connection step with the derivation of the link key resp. long-term key as a separate session (creating an empty session key but initializing a permanent connection key), even though usually computation of an encryption key would immediately follow the initial connection. We let the adversary decide when and how often devices reconnect.

The TOFU property indicates if the session key should be considered to be secure. When initializing a new session, we declare this session to be not trustworthy, and only change this later if there is an honest partner session to which the session here is connected to, i.e., if the adversary has been passive. All subsequent reconnections of the session then inherit this flag. Overall, we thus have three flags for keys: $\mathsf{isTested}$ for session keys that have been tested, $\mathsf{isRevealed}$ for session keys that have been revealed, and isTOFU for session keys that have been derived from a trustworthy initialization step. The latter flags refine the usual freshness condition for session keys.

3.1 Attack Model

We give a game-based security model in the Bellare–Rogaway style [BR94]. We assume that parties have some identity. For Bluetooth, this will be the 48-bit Bluetooth device address `BD_ADDR`, which can be either public or random. In accordance to the Bluetooth protocol description, we sometimes denote the identities of connecting devices as A and B. Parties know their identity and also learn the intended partner's id when the cryptographic protocol starts (via device discovery). We note that Bluetooth addresses can be easily changed on a device and are usually not authenticated.

As explained in the introduction, we are interested in the trust-on-first-use security of the protocol. We model this by declaring a trustworthy relationship if two sessions of honest parties are partnered, indicating that the adversary has been passive in the initial connection. From then on, the (now active) adversary can interact with either of the two parties. We note that the adversary can still start initial connections with any party and actively participate in these connection. We do not aim to protect the session keys in such connections but since parties may reuse secret information like the Diffie–Hellman shares in multiple executions, we need to account for such attack vectors.

For the reusable Diffie–Hellman key, we assume that each party i, at the beginning of the game, is initialized with a key pair $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KGen}(1^\lambda)$. To model that the key may or may not be used in several sessions, we grant the adversary access to a $\mathsf{NextPK}(i)$ oracle, which renews the key pair of party i. We note that the new key pair will only be used in future sessions, not in the currently running ones. This means that each session is assigned a unique key pair. This is modeled by having a counter value pkctr_i, initialized to 0, which is incremented with each key rolling.

Sessions. A protocol session $\mathsf{lbl} = (i, k)$ is given by a pair consisting of the k-th session in a protocol run of party with identity i. When the adversary initiates a new session, the game assigns the next available integer k. Each such session lbl holds a set of entries:

- id is the identity i of the party.
- mode, either init or $\mathsf{reconnect}$, describes if this is a new initial connection or a reconnection.

- aux denotes some auxiliary information like the association model JustWorks, PasskeyEntry, NumCom, or OOB that should be used, and further data like $\mathsf{passkey} \in \{0, 1, \ldots, 9\}^* \cup \{\perp\}$ in the PasskeyEntry subprotocol, or information transmitted out-of-band.

- $\mathsf{ConnectKey}$ describes the connection key (called link key in Bluetooth Classic and long-term key in Bluetooth Low Energy), which is set during the initial connection and used later to derive further session keys when reconnecting. Initialized to $\perp$.

- The variable state determines if the session is $\mathsf{running}$, or has $\mathsf{accepted}$ or $\mathsf{rejected}$.

- The Boolean variable $\mathsf{isTested}$ determines if the session key has been tested before. Initialized to false.

- The Boolean variable $\mathsf{isRevealed}$ defines if the session has been revealed. Initialized to false.

- The Boolean variable isTOFU determines if the session key has been derived following a trustworthy initial connection. Initialized to false.

- pkctr denotes the counter value of key pair used by party i in the session. When performing protocol steps, the party always uses the key pair identified by this counter value. But the party may actually use different keys in different sessions concurrently.

- $\mathsf{key} \in \{0, 1\}^* \cup \{\perp\}$ describes the session key, initialized to $\perp$. Note that for a successful initial connection in Bluetooth, the session key remains empty, i.e., is set to $\perp$.

- $\mathsf{sid} \in \{0, 1\}^* \cup \{\perp\}$ is the session identifier, the initial value is $\perp$. The session identifier is set only once during an execution.

A central property in key-exchange protocols is to define when two sessions belong to each other. We use here the common approach to say that two (distinct) sessions are partnered if they hold the same (non-trivial) session identifier:

Definition 3.1 (Partnered Sessions) *We say that two sessions* $\mathsf{lbl}, \mathsf{lbl}'$ *are partnered if* $\mathsf{lbl} \neq \mathsf{lbl}'$ *and* $\mathsf{lbl}.\mathsf{sid} = \mathsf{lbl}'.\mathsf{sid} \neq \perp$*.*

Note that $\mathsf{sid} \neq \perp$ presumes that the session has accepted.

Adversarial Queries. We consider an active adversary $\mathcal{A}$ interacting with the protocol. The adversary has an access to the following oracle queries:

- $\mathsf{InitSession}(i, [\mathsf{aux}])$ establishes a new session at party i (with number k). Assigns the corresponding values to the entries in $\mathsf{lbl} = (i, k)$, i.e., $\mathsf{lbl}.\mathsf{id} \leftarrow i$, the mode is set to $\mathsf{lbl}.\mathsf{mode} \leftarrow \mathsf{init}$, and the optional parameter $[\mathsf{aux}]$, if present, is stored in $\mathsf{lbl}.\mathsf{aux}$ (and otherwise this entry is set to $\perp$). We set $\mathsf{lbl}.\mathsf{state} \leftarrow \mathsf{running}$, $\mathsf{lbl}.\mathsf{pkctr} \leftarrow \mathsf{pkctr}_i$, as well as $\mathsf{lbl}.\mathsf{isTested}, \mathsf{lbl}.\mathsf{isRevealed}, \mathsf{lbl}.\mathsf{isTOFU} \leftarrow \mathsf{false}$, since this establishes a new session in which the active adversary may interact with party i. Return lbl.

- $\mathsf{Reconnect}(\mathsf{lbl}, [\mathsf{aux}])$ checks if there exists a session with label lbl with $\mathsf{lbl}.\mathsf{ConnectKey} \neq \perp$. If so, it establishes a new session $\mathsf{lbl}' = (i, k')$ via calling $\mathsf{InitSession}(i, [\mathsf{aux}])$ but immediately overwrites $\mathsf{lbl}'.\mathsf{mode} \leftarrow \mathsf{reconnect}$. The new session inherits the TOFU characteristic of the preceding session, that is, one sets $\mathsf{lbl}'.\mathsf{isTOFU} \leftarrow \mathsf{lbl}.\mathsf{isTOFU}$, and duplicates the previous connection key, $\mathsf{lbl}'.\mathsf{ConnectKey} \leftarrow \mathsf{lbl}.\mathsf{ConnectKey}$. Return lbl'.

- $\mathsf{Send}(\mathsf{lbl}, m)$ sends a protocol message m to the session lbl. Returns $\perp$ if the session does not exist or is not established, and the party's protocol reply otherwise. When executing the command, the protocol party may set $\mathsf{lbl.sid}$ or change the state $\mathsf{lbl.state}$ to $\mathsf{accepted}$ or $\mathsf{rejected}$. If $\mathsf{lbl.state}$ turns to $\mathsf{accepted}$ then check the following:
  - If $\mathsf{lbl.mode} = \mathsf{init}$ and there exists a partnered session lbl' to lbl then set $\mathsf{lbl.isTOFU} \leftarrow \mathsf{true}$ and $\mathsf{lbl}'.\mathsf{isTOFU} \leftarrow \mathsf{true}$.
  - If there exists a partnered session lbl' with $\mathsf{lbl}'.\mathsf{isTested} = \mathsf{true}$ then set $\mathsf{lbl.isTested} \leftarrow \mathsf{true}$. This mirrors the property for partnered sessions.
  - If there exists a partnered session lbl' with $\mathsf{lbl}'.\mathsf{isRevealed} = \mathsf{true}$ then set $\mathsf{lbl.isRevealed} \leftarrow \mathsf{true}$.
- $\mathsf{NextPK}(i)$ updates the key pair of party i. That is, increment pkctr_i and compute a new key pair $(\mathsf{sk}_i[\mathsf{pkctr}_i], \mathsf{pk}_i[\mathsf{pkctr}_i]) \leftarrow \mathsf{KGen}(1^\lambda)$.
- $\mathsf{Reveal}(\mathsf{lbl})$ returns the session key key of session lbl, or $\perp$ if the session does not exist, or if $\mathsf{lbl.state} \neq \mathsf{accepted}$, or if $\mathsf{lbl.isRevealed} = \mathsf{true}$. Sets $\mathsf{lbl.isRevealed} \leftarrow \mathsf{true}$ and also $\mathsf{lbl}'.\mathsf{isRevealed} \leftarrow \mathsf{true}$ for all partnered sessions lbl' with $\mathsf{lbl}'.\mathsf{sid} = \mathsf{lbl.sid}$.
- $\mathsf{Test}(\mathsf{lbl})$ tests the session key key of the session lbl. If the session does not exist, or $\mathsf{lbl.isRevealed} = \mathsf{true}$, or $\mathsf{lbl.isTOFU} = \mathsf{false}$, or $\mathsf{key} = \perp$, or $\mathsf{lbl.state} \neq \mathsf{accepted}$, or $\mathsf{lbl.isTested} = \mathsf{true}$, then immediately returns $\perp$. Else returns either the real key key or a random string of length $|\mathsf{key}|$, depending on the random bit b chosen by the challenger $\mathcal{C}$. Sets $\mathsf{lbl.isTested} \leftarrow \mathsf{true}$ to make sure that the adversary potentially does not get another random key when testing this session again. For the same reason, it also sets $\mathsf{lbl}'.\mathsf{isTested} \leftarrow \mathsf{true}$ for all partnered sessions lbl' with $\mathsf{lbl}'.\mathsf{sid} = \mathsf{lbl.sid}$.

When considering attacks against the Bluetooth protocol, we assume a set $\mathcal{I}$ of admissible identities. We denote by $\mathcal{L}$ the set of session labels lbl activated by the adversary.

3.2 Security Properties

We state the two common security properties of key-exchange protocols. One is Match-security, covering basic functional guarantees such as honest executions deriving the same session key, and that the partnering condition is not "too loose". The other one is key secrecy. We note that we often define the properties in the asymptotic sense for sake of simplicity. But we give concrete security bounds when analyzing the Bluetooth security suite.

In the definition, we give the adversary access to the same oracles as for key secrecy, e.g., including a Test oracle, albeit not oracles may be relevant for the attack. This is only to unify both attacks.

Match-Security. Intuitively, Match-security states that, if two sessions are partnered then they also hold the same session key (1), and at most two sessions are partnered (2). For reconnections, the former should only hold for sessions, which have been connected before and thus hold the same connection key. We therefore stipulate that the $\mathsf{ConnectKey}$-entry in both executions must be identical if one of the sessions is in mode $\mathsf{mode} = \mathsf{reconnect}$, and split the first requirement into one for initial connections (if at least one party is in mode $\mathsf{mode} = \mathsf{init}$) and one for reconnections.

Definition 3.2 (Match-Security) *We say that a KE protocol* Π *provides Match-security if for any PPT adversary* $\mathcal{A}$ *and identity set* $\mathcal{I}$*, we have*

$$\boldsymbol{Adv}^{\mathit{Match}}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda) := \Pr\left[\boldsymbol{Exp}^{\mathit{Match}}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda) = 1\right]$$

is negligible, where

```
$\boldsymbol{Exp}^{Match}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda)$
$b \leftarrow_{\$} \{0,1\}$
$\textbf{\textit{forall}}\ i \in \mathcal{I}$ do
    $\mathsf{pkctr}_i \leftarrow 0$
    $(\mathsf{sk}_i[0], \mathsf{pk}_i[0]) \leftarrow_{\$} \mathsf{KGen}(1^\lambda)$
$\mathcal{A}^{\mathsf{InitSession},\mathsf{Reconnect},\mathsf{Send},\mathsf{NextPK},\mathsf{Reveal},\mathsf{Test}}(\{(i, \mathsf{pk}_i[0])\}_{i \in \mathcal{I}})$
return 1 if
  $\exists$ pairwise distinct $\mathsf{lbl}, \mathsf{lbl}', \mathsf{lbl}'' \in \mathcal{L}$ :
    $(1a)$ $\mathsf{lbl.sid} = \mathsf{lbl}'.\mathsf{sid} \neq \bot$ and $\mathsf{lbl.mode} = \mathsf{init}$ and $\mathsf{lbl.key} \neq \mathsf{lbl}'.\mathsf{key}$
    $(1b)$ $\mathsf{lbl.sid} = \mathsf{lbl}'.\mathsf{sid} \neq \bot$ and $\mathsf{lbl.mode} = \mathsf{reconnect}$ and $\mathsf{lbl.ConnectKey} = \mathsf{lbl}'.\mathsf{ConnectKey}$ and $\mathsf{lbl.key} \neq \mathsf{lbl}'.\mathsf{key}$
    $(2)$ $\mathsf{lbl.sid} = \mathsf{lbl}'.\mathsf{sid} = \mathsf{lbl}''.\mathsf{sid} \neq \bot$
```

Key Secrecy. Next we define what it means that a session key, derived after a trustworthy initialization step, remains secret. This should hold even if the adversary mounts an active attack after the TOFU step. We note that we only need to check eventually that no session has been tested and revealed (or its partner session has been revealed). The TOFU property, that only keys that have been created in a trustworthy way should be kept secret, is ensured by the attack model (e.g., the Test oracle immediately rejects requests for session keys with isTOFU = false).

Definition 3.3 (Key Secrecy) *We say that a key exchange protocol* Π *provides Secrecy if for any PPT adversary* $\mathcal{A}$ *and identity set* $\mathcal{I}$ *we have*

$$\boldsymbol{Adv}^{Secrecy}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda) := \Pr\left[\boldsymbol{Exp}^{Secrecy}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda) = 1\right] - \frac{1}{2}$$

is negligible, where

```
$\boldsymbol{Exp}^{Secrecy}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda)$
$b \leftarrow_{\$} \{0,1\}$
$\textbf{\textit{forall}}\ i \in \mathcal{I}$ do
    $\mathsf{pkctr}_i \leftarrow 0$
    $(\mathsf{sk}_i[0], \mathsf{pk}_i[0]) \leftarrow_{\$} \mathsf{KGen}(1^\lambda)$
$a \leftarrow_{\$} \mathcal{A}^{\mathsf{InitSession},\mathsf{Reconnect},\mathsf{Send},\mathsf{NextPK},\mathsf{Reveal},\mathsf{Test}}(\{(i, \mathsf{pk}_i[0])\}_{i \in \mathcal{I}})$
return 1 if
$a = b$ and there are no sessions $\mathsf{lbl}, \mathsf{lbl}' \in \mathcal{L}$ with
        $\mathsf{lbl.sid} = \mathsf{lbl}'.\mathsf{sid}$ but $\mathsf{lbl.isRevealed} = \mathsf{false}$ and $\mathsf{lbl}'.\mathsf{isTested} = \mathsf{true}$
```

4 Security of Bluetooth

In this section, we show that the Bluetooth protocol suite (for both BR/EDR and BLE) constitutes a secure TOFU key-exchange protocol. In the security statements below, we usually refer to the Bluetooth protocol Π, capturing either $\Pi_{\mathrm{BR/EDR}}$ or Π_{BLE}, and only refine the concrete security bounds with respect to the specific protocol transport. We note that we view the initial pairing phase as creating a permanent key, equal to the link key resp. long-term key, but formally no session key. Session keys are then derived via the corresponding mechanisms in the protocol. This is valid since the model also allows empty session keys, which trivially satisfy correctness and security properties.

4.1 Security Assumptions

For our security results, we merely need two assumptions. One is the PRF-ODH assumption to draw conclusions about the reused Diffie–Hellman value in the SSP protocol, and the other one is the key derivation in the reconnection steps.

PRF-ODH Assumption. The PRF-ODH assumption states that applying a pseudorandom function PRF to a Diffie–Hellman key g^{uv} and an adversarial chosen string x^* looks random, even if the adversary learns related outputs of PRF. The only restriction is that the adversary cannot ask for $\mathsf{PRF}(g^{uv}, x^*)$ directly. We work here with the so-called mm setting [BFGJ17] where the adversary can make *m*ultiple queries for both Diffie–Hellman keys g^u and g^v. This is necessary since either Bluetooth device may reuse the key in other sessions. We also assume that the adversary has access to both Diffie–Hellman parts and oracles at the outset.

We formulate the PRF-ODH assumption close to the deployment in Bluetooth. The Bluetooth specification [BT523, Vol 2, Part H, Section 7.6] says that the secret exponents shall be picked between 1 and $q/2$. This does not seem to violate the security of the PRF-ODH assumption, since the most significant bit of the discrete logarithm is a hardcore bit [BM84, HN04]. Hence the adversary's behavior should not change significantly when given g^u with u being in the range 1 and $q/2$, or in the range $q/2$ and q. Analogously for v.

Further, the key derivation only uses the x-coordinate $[g^{ab}]_x$ of the Diffie–Hellman value shared by the two parties. Note that this coordinate determines the entire value up to the sign of the y-coordinate, and that the two options for the sign describe the value g^{ab} and its inverse (written as g^{-ab} in multiplicative form). We modify the PRF-ODH accordingly by using only the x-coordinate in the computations. This causes us to be more restrictive when it comes to the queries to the related values, implemented via the oracles ODH_u and ODH_v: We require that the adversary never queries these oracles about the challenge value g^{uv} with label x^*, nor about g^{-uv} with x^* (because both values would yield the same output after projection of g^{uv} resp. g^{-uv} onto the same x-coordinate). We note that in the security proof for Bluetooth, our reduction to the PRF-ODH problem will indeed only use different labels $x \neq x^*$ for ODH queries, such that this is always satisfied.

Definition 4.1 (PRF-ODH Assumption) *Let $\mathbb{G}$ be a cyclic group of prime order $q = q(\lambda)$ generated by g. Let $\mathsf{PRF} : \mathbb{G} \times \{0,1\}^* \to \{0,1\}^*$ be a pseudorandom function, taking a key $k \in \mathbb{G}$ and a string s as input and producing a string $\mathsf{PRF}(k, s)$ as output. For a given $w \in \mathbb{Z}_q$, let $\mathsf{ODH}_w : \mathbb{G} \times \{0,1\}^* \to \{0,1\}^*$ be the function, which takes as input $X \in \mathbb{G}$ and string s and returns $\mathsf{PRF}([X^w]_x, s)$.*

We say that the PRF-ODH assumption holds relative to $\mathbb{G}$ if for any PPT adversary $\mathcal{A}$, we have

$$\boldsymbol{Adv}^{\textit{PRF-ODH}}_{\mathcal{A},\mathsf{PRF},\mathbb{G}}(\lambda) := \Pr\left[\boldsymbol{Exp}^{\textit{PRF-ODH}}_{\mathcal{A},\mathsf{PRF},\mathbb{G}}\right] - \tfrac{1}{2}$$

is negligible, where

```
$\boldsymbol{Exp}^{\textit{PRF-ODH}}_{\mathcal{A},\mathsf{PRF},\mathbb{G}}$
$u, v \leftarrow\$ \mathbb{Z}_{(q+1)/2} \setminus \{0\}, b \leftarrow\$ \{0,1\}$
$U \leftarrow g^u, V \leftarrow g^v$
$(x^*, \mathsf{st}) \leftarrow\$ \mathcal{A}^{\mathsf{ODH}_u(\cdot,\cdot),\mathsf{ODH}_v(\cdot,\cdot)}(U, V)$
$y_0 \leftarrow \mathsf{PRF}([g^{uv}]_x, x^*), y_1 \leftarrow \{0,1\}^{|y_0|}$
$a \leftarrow\$ \mathcal{A}^{\mathsf{ODH}_u(\cdot,\cdot),\mathsf{ODH}_v(\cdot,\cdot)}(\mathsf{st}, V, y_b)$
**return** $a = b$
```

where we assume that $\mathcal{A}$ never makes a query $(A, x) = (V^{\pm 1}, x^*)$ to oracle ODH_u resp. $(B, x) = (U^{\pm 1}, x^*)$ to ODH_v.

We note that for Bluetooth Classic, the pseudorandom function $\mathsf{PRF}(W,x)$ is $\mathsf{HMAC}(W,x)$. For BLE, it is a nested CMAC computation, $\mathsf{PRF}(W,x) = \mathsf{CMAC}(\mathsf{CMAC}(\text{Salt}, W), x)$. It seems plausible to assume that the PRF-ODH assumption holds for these instantiations. We also note that the PRF-ODH assumption implicitly stipulates that the Diffie–Hellman problem is hard, i.e., small subgroup attacks such as in [BN19] must be prevented. This is usually done by checking the validity of the curve points.

Pseudorandom Function. For the reconnection steps, we require that the underlying function HMAC in BR/EDR and AES in BLE, from which the encryption keys are derived, behave like pseudorandom functions. For an adversary $\mathcal{C}$, let $\mathbf{Adv}^{\mathrm{PRF}}_{\mathcal{C},\mathsf{PRF}}(\lambda)$ denote the common security advantage of $\mathcal{C}$ distinguishing a $\mathsf{PRF}(k,\cdot)$ oracle from a random-function oracle, the choice what oracle is used made at random.

4.2 Match Security

We first argue Match Security of the Bluetooth protocol. Recall that we set the session identifiers for the initial connection to consist of $\mathsf{sid} = ([g^a]_x, [g^b]_x, A, B, Na, Nb)$, for BR/EDR reconnections of $\mathsf{sid} = (AU\_RAND\_C, AU\_RAND\_P, A, B)$ resp. for BLE reconnections of $\mathsf{sid} = (SKD\_C, SKD\_P)$. Also note that the parties may reuse their Diffie–Hellman secret across multiple executions; the nonces, however, are fresh 128-bit values, chosen randomly in each session and present in each of the SSP subprotocols. Furthermore, recall that the initial connection derives an empty session key and that the link key resp. long-term key is stored as the permanent key in entry $\mathsf{ConnectKey}$ of the session.

Proposition 4.2 (Match Security) *The Bluetooth protocol* Π *provides Match security. That is, for any adversary* $\mathcal{A}$ *calling at most* q_s *sessions, we have*

$$\boldsymbol{Adv}^{Match}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda) \leq q_s^2 \cdot 2^{-|nonce|},$$

where $|nonce| = 128$ *for* BR/EDR *and* $|nonce| = 64$ *for* BLE*.*

The reason for having different bounds stems from the distinct key derivation when reconnecting. Both protocol versions use 128-bit nonces for initial connection, but only BR/EDR uses 128 bit values for reconnections; BLE instead uses the 64-bit session key diversifiers.

Proof. For the first properties, (1a) and (1b), that partnered sessions have the same session key, note that the link/long-term key in an initial connection is computed as $\mathsf{KDF}([g^{ab}]_x, Na, Nb, A, B)$ such that the output of the (deterministic) key derivation matches for equal session identifiers. Recall that the x-coordinates of g^a and g^b determine the one of g^{ab}. Also, session identifiers for the initial connection and reconnections differ in length such that they cannot match the other type (in both BR/EDR and BLE). For reconnections, the session identifiers $(AU\_RAND\_C, AU\_RAND\_P, A, B)$ resp. (SKD_C, SKD_P) fully specify the derived session keys together with the same link/long-term key, implying a match as well.

For the second property, note that if there were three sessions with the same session identifier sid, then two of them must be in the role of Alice (or Bob). If we have at most q_s sessions in total, there are at most q_s^2 such pairs of two Alice- or Bob-sessions. The honest party picks a fresh nonce Na resp. Nb in each of these two executions (for initial connections in either mode) and fresh values AU_RAND_C, AU_RAND_P for reconnections in BR/EDR resp. 64-bit values SKD_C, SKD_P in BLE. It follows that each pairs yields a nonce collision with probability at most $2^{-|\text{nonce}|} = 2^{-128}$ in BR/EDR resp. $\leq 2^{-64}$ in BLE. The overall threefold collision probability for session identifiers is thus at most $q_s^2 \cdot 2^{-|\text{nonce}|}$ as stated. $\square$

4.3 Key Secrecy

As it turns out, key secrecy does not depend on the Authentication stages 1 and 2 of the protocol. As such, the analysis easily works for all modes of the protocol simultaneously.

Proposition 4.3 (Key Secrecy) *The Bluetooth protocol* Π *provides trust-on-first-use Key secrecy. That is, for any adversary* $\mathcal{A}$ *initiating at most* $q_s = q_{init} + q_{rec}$ *sessions (of which* q_{init} *are initial connections and* q_{rec} *are reconnections), there exist efficient adversaries* $\mathcal{B}$ *and* $\mathcal{C}$ *(with roughly the same run time as* $\mathcal{A}$*), such that*

$$\boldsymbol{Adv}^{Secrecy}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda) \leq q^2_{init} \cdot 2^{-|128|} + q^3_{init} \cdot \boldsymbol{Adv}^{PRF\text{-}ODH}_{\mathcal{B},\mathsf{PRF},\mathbb{G}}(\lambda) + [2]_{\mathrm{BR/EDR}} q_{rec} \cdot \boldsymbol{Adv}^{PRF}_{\mathcal{C},\mathsf{PRF}'}(\lambda) + [q^2_{rec} \cdot 2^{-64}]_{\mathrm{BR/EDR}}.$$

where PRF *in the PRF-ODH case is* HMAC *for* BR/EDR *resp.* $\mathsf{CMAC}(\mathsf{CMAC}(\mathit{Salt}, \cdot), \cdot)$ *for* BLE*, and* PRF' *for reconnections is* HMAC *for* BR/EDR *resp.* AES *for* BLE*, each with keys of* 128 *bits. We note that the term* $q^2_{rec} \cdot 2^{-64}$ *only appears for* BR/EDR *(what is indicated by the notation* $[\ ]_{\mathrm{BR/EDR}}$*), as well as one of the terms* $q_{rec} \cdot \boldsymbol{Adv}^{PRF}_{\mathcal{C},\mathsf{PRF}'}(\lambda)$ *because of the additional key computation in* BR/EDR*.*

We note that the reduction factor q^3_{init} is indeed large but follows other analyses. A factor q_{init} comes from the multiple test queries, which our model allows, and the quadratic term q^2_{init} from the need to guess the correct insertion points of the Diffie–Hellman keys. For instance, Troncoso and Hale [TH21] also have the quadratic loss factor for the model with a single-test query. Tighter security bounds usually require other techniques as used in Bluetooth [GJ18] or to use and program a random oracle [CCG+19, DJ21]. The latter may nonetheless be a viable way to reduce the loss factor in Bluetooth as well. On the other hand, since Bluetooth is a short-range technique, mounting attacks with an extensive number of sessions seems to be hard. Indeed, a factor q^2_{init} would disappear if the adversary had to announce the target in advance.

Proof. The proof proceeds via game hopping. We start with the original attack on the Bluetooth protocol. Then we gradually change the game till we reach the point where, independently of the challenge bit b, the adversary only gets to see random keys. We denote by $\Pr[\mathsf{Game}_j]$ the probability that the adversary wins in the corresponding game (over the guessing probability). In particular, $\Pr[\mathsf{Game}_0] = \mathbf{Exp}^{Secrecy}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda) - \frac{1}{2}$.
Game 0. Is the original attack on the protocol. We assume in the following without loss of generality that the adversary never reveals or tests an empty session key of a session in mode $\mathsf{mode} = \mathsf{init}$.
Game 1. In Game_1, we assume that there are no three sessions (in mode $\mathsf{mode} = \mathsf{init}$) with the same session identifier.

It follows as in the case of Match security that this happen with probability at most $q^2_{init} \cdot 2^{-|\text{nonce}|}$. Note that we here have $|\text{nonce}| = 128$ (and not 64) because both versions, BR/EDR and BLE, use 128-bit nonces in the pairing step.
Game 2. In Game_2, we replace the connection key $\mathsf{ConnectKey}$ in each session lbl in mode $\mathsf{lbl.mode} = \mathsf{init}$ upon acceptance as follows: If there is a partnered session lbl' that has accepted before—there can be at most one by the previous game hop—set $\mathsf{lbl.ConnectKey} \leftarrow \mathsf{lbl}'.\mathsf{ConnectKey}$. Else, replace $\mathsf{lbl.ConnectKey}$ by a random string of the same length. Any other connection key is computed according to the protocol.

Observe that the sessions where we replace keys are those, which are considered to be trustworthy in the sense that they completed an initial execution with a passive adversary ($\mathsf{isTOFU} = \mathsf{true}$). We note that the former step in the replacement above only ensures consistency; in the protocol execution in Game_1, the parties would derive the same $\mathsf{ConnectKey}$ by construction.

We argue that $\Pr[\mathsf{Game}_1] \leq \Pr[\mathsf{Game}_2] + q^3_{\text{init}} \cdot \mathbf{Adv}^{\text{PRF-ODH}}_{\mathcal{B},\mathsf{PRF},\mathbb{G}}(\lambda)$. The argument is via an (interactive) hybrid argument against the PRF-ODH assumption. That is, our adversary $\mathcal{B}$ picks a random index h between 1 and q_{init} and replaces (in a consistent way) the first $h-1$ such keys randomly, for the h-th key $\mathcal{B}$ uses the challenge value of the PRF-ODH game, and leaves the remaining keys untouched (unless it overwrites them for consistency reasons). To map at most q_{init} Diffie–Hellman keys to the PRF-ODH challenge values U, V, our algorithm $\mathcal{B}$ try to guess the right indexes i, j of the insertion positions of U, V as responses to NextPK calls. If the predictions for i or j turn out to be wrong, we will output a random guess instead, effectively reducing the distinguishing advantage against PRF-ODH by a factor of q_{init}^2.

Adversary $\mathcal{B}$ runs $\mathcal{A}$'s attack in Game_1, simulating the protocol steps of the honest parties genuinely and with the replacement explained above according to the hybrid parameter h. Note that $\mathcal{B}$ can use its oracles ODH_u and ODH_v to perform the key derivation and MAC-key computation on behalf of the honest party for other sessions involving the Diffie–Hellman parameters U, V, without knowing the discrete logarithms. That is, for any received group element C and label x (specified by the Bluetooth protocol) in a protocol execution for the party's Diffie–Hellman share U or V, our adversary calls its oracle about (C, x) to compute the correct value. We argue below that the queries also comply with the restriction on trivial queries to ODH_u and ODH_v.

Our adversary $\mathcal{B}$ eventually replaces the first $h - 1$ connection keys by random values (but consistently). For the h-th connection key, algorithm $\mathcal{B}$ calls its PRF-ODH challenge oracle about label $x^* = Na|Nb|\texttt{'}btlk\texttt{'}|A|B$ resp. $x^* = \texttt{0x01}|\texttt{'}btle\texttt{'}|Na|Nb|A|B|\texttt{0x0100}$ according to the protocol description. Algorithm $\mathcal{B}$ overwrites the connection key with the answer to the challenge (unless the key has already been overwritten for consistency reasons). The remaining connection keys remain unchanged, unless they need to be adapted for consistency reasons.

We note that the nonces are unique in two sessions such that the label x^* asked to the challenger in the PRF-ODH game can only appear later again if the partnered session completes its execution. In this case, our algorithm $\mathcal{B}$ can simply copy the $\mathsf{ConnectKey}$ value. For any other label $x \neq x^*$, it can query its oracles ODH_u and ODH_v to derive the same keys.

Adversary $\mathcal{B}$ eventually checks if $\mathcal{A}$ succeeds in attacking the game. If so, it outputs its guess 0, else 1 (or a random bit if the predictions for i or j have been wrong). A standard analysis of the hybrid method shows that the difference between the probabilities in Game_1 and Game_2 is bounded by $\mathcal{B}$'s advantage against the PRF-ODH assumption (times the loss factor q_{init}^3).

BR/EDR reconnections. For the reconnection step, we branch into two cases, one for BR/EDR (Games Game_{3a} to Game_{5a}) and one for BLE (Game Game_{3b}). However, the bounds are almost equal up to a factor 2 for BR/EDR, such that we can subsume both cases in the final bound.

Game 3a. In any session lbl of mode $\mathsf{reconnect}$ with $\mathsf{isTOFU} = \mathsf{true}$, consistently replace the computation $\mathsf{HMAC}(LK, \dots)/2^{128}$ (for both device key derivation as well as session key derivation) by random values. Consistently means that fresh inputs, either for the key or the value, yield fresh random outputs, but repeating inputs return the previous answer.

By the pseudorandomness of HMAC, we can easily give a reduction, simulating the game for the adversary with access to at most q_{rec} function oracles (which we can formally reduce to a single oracle via a hybrid argument), each oracle queried at most q_{rec} times. In the simulation, we identify the right oracle instance via the session identifiers in the initial pairing step. We get that

$$\Pr[\mathsf{Game}_2] \leq \Pr[\mathsf{Game}_{3a}] + q_{\text{rec}} \cdot \mathbf{Adv}^{\text{PRF}}_{\mathcal{C},\mathsf{HMAC}}(\lambda).$$

Note that computations for the device key and for the session keys take inputs of different lengths and thus always yields independent random outputs.

Game 4a. In any session lbl of mode $\mathsf{reconnect}$ with $\mathsf{isTOFU} = \mathsf{true}$, (consistently) replace the values $SRES\_C|SRES\_P|\text{ACO} \leftarrow \mathsf{HMAC}(\text{dk}, AU\_RAND\_C|AU\_RAND\_P)/2^{128}$ by random values.

Since dk is already random, we can conclude once more from the pseudorandomness of HMAC that

$$\Pr[\mathsf{Game}_{3a}] \leq \Pr[\mathsf{Game}_{4a}] + q_{\text{rec}} \cdot \mathbf{Adv}^{\text{PRF}}_{\mathcal{C},\mathsf{HMAC}}(\lambda).$$

Here we use for simplicity the same distinguisher name $\mathcal{C}$ as in the previous step, implicitly taking the maximum for both distinguishers and yielding a factor 2 in the final bound.

Game 5a. Let the adversary lose if there are two accepting sessions lbl and lbl', both in mode reconnect and with $\mathsf{isTOFU} = \mathsf{true}$, which are not partnered but such that the values ACO and $\mathtt{BD\_ADDR}_A|\mathtt{BD\_ADDR}_B$ are identical in both sessions.

Assume that there are two sessions lbl and lbl' as above. The sessions are not partnered but have the same Bluetooth addresses. Since the session identifier is given by $\mathsf{sid} = (AU\_RAND\_C, AU\_RAND\_P, \mathtt{BD\_ADDR}_A, \mathtt{BD\_ADDR}_B)$, we must have that the nonces (AU_RAND_C, AU_RAND_P) in session lbl and (AU_RAND_C', AU_RAND_P') in session lbl' must be distinct. The values $SRES\_C|SRES\_P|\text{ACO}$ and $SRES\_C'|SRES\_P'|\text{ACO}'$ in the two sessions are thus picked randomly and independently according to the previous game, in which we replaced outputs for $\mathsf{HMAC}(\mathrm{dk}, \cdot)/2^{128}$ consistently by random values. We therefore have a collision on the ACO-part with probability at most 2^{-64} for such sessions. Since we have at most q_{rec}^2 reconnection session pairs $\mathsf{lbl}, \mathsf{lbl}'$, we get

$$\Pr[\mathsf{Game}_{4a}] \leq \Pr[\mathsf{Game}_{5a}] + q_{\text{rec}}^2 \cdot 2^{-64}.$$

In summary, the final session keys $\mathsf{k}_{\text{AES}} \leftarrow \mathsf{HMAC}(LK,' btak'|\mathtt{BD\_ADDR}_A|\mathtt{BD\_ADDR}_B|\text{ACO})/2^{128}$ for reconnection sessions with $\mathsf{isTOFU} = \mathsf{true}$ are all random and independent (unless two such sessions are partnered, in which case the keys are random but consistent). It follows that any adversary always gets a fresh random session key on testable sessions (noting that only one session of two partnered sessions can be tested in our model), and thus has a prediction probability of the test bit b of exactly $\frac{1}{2}$ to distinguish such session keys in Game_{5a} from random answers.

BLE reconnections. It remains to investigate the bounds for BLE reconnections. We start again with Game_2 and hop to Game_{3b} now.

Game 3b. In any session lbl of mode reconnect with $\mathsf{isTOFU} = \mathsf{true}$, consistently replace the computation $\mathsf{AES}(LTK, \dots)$ (for session key derivation) by random values. Consistently means that fresh inputs, either for the key or the value, yield fresh random outputs, but repeating inputs return the previous answer.

By the pseudorandomness of AES, we can analogously to BR/EDR and HMAC give a reduction to the pseudorandomness of AES, specifically to at most q_{rec} oracles, each queried at most q_{rec} times. Once more, we identify the right oracle instance via the session identifiers in the initial pairing step. We get that

$$\Pr[\mathsf{Game}_2] \leq \Pr[\mathsf{Game}_{3b}] + q_{\text{rec}} \cdot \mathbf{Adv}_{\mathcal{C},\mathsf{AES}}^{\text{PRF}}(\lambda).$$

We would now make another hop to a game Game_{4b} in which we made sure, analogously to Game_{5a} in the BR/EDR case, that key derivation (for the same LTK) uses fresh inputs SKD_C, SKD_P across sessions—unless these sessions are already partnered. However, note here that this is given by construction already, since the pair SKD_C, SKD_P already forms the session identifier. It follows here, too, that reconnection session with $\mathsf{isTOFU} = \mathsf{true}$ are all random and independent (unless two such sessions are partnered, in which case, the keys are random but consistent).

We can now take the advantages of both cases and add them up to derive the overall bound. □

5 Privacy in Bluetooth LE

Bluetooth Low Energy supports address randomization technique to provide privacy. We show here that this mechanism indeed achieves privacy (against outsiders) if one neglects other attack possibilities based on physical features or other observable data.

5.1 Details on Privacy Mechanisms in Bluetooth Low Energy

For the BLE protocol, we dive into the link establishment process to understand better the privacy mechanisms.

Types of Addresses. To support the privacy mechanism, the standard specifies four types of Bluetooth addresses `BD_ADDR` (their relation is given in Figure 7) in LE:

Public Addresses: A globally unique device identifier `MAC`, consisting of a 24-bit vendor identifier and a local identifier chosen by the vendor.

Static Random Address: A random address, which is set once for the device's lifetime or can be changed upon reboots. Such addresses carry the most significant bit values '11', what allows distinguishing them from the next two types.

Non-Resolvable Random Private Addresses: A frequently changed random address (with the most significant bits set to '00'). The standard recommends to renew random addresses, including this type and the next one, at least every 15 minutes [BT523, Vol 3, Part C, App. A].

Resolvable Random Private Addresses: A random address wherefrom a trusted device can extract the Public or Static Random Addresses. It consists of 24 bits constituting *prand* that are set randomly—effectively only 22 random bits since the most significant bits correspond to '01'— and the other 24 bits are computed as a (pseudorandom) hash from *prand* for an Identity Resolving Keys (*IRK*). This Identity Resolving Key must have been shared with the trusted device in a previous connection.

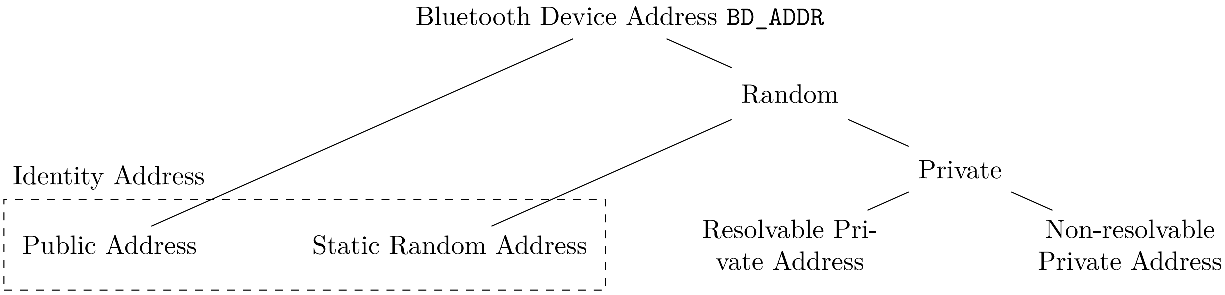
Bluetooth Device Address BD_ADDR
Random
Private
Identity Address
Public Address
Static Random Address
Resolvable Private Address
Non-resolvable Private Address

Figure 7: Types of the Bluetooth Device Addresses in BLE.

Generating Resolvable Random Private Addresses. The Identity Resolving Key *IRK* is a device-specific 128-bit value. It can be assigned or generated randomly during manufacturing, but the standard also allows any other methods to create the *IRK*. It can be also generated from a 128-bit Identity Root *IR* as $IRK \leftarrow \mathsf{AES}(IR, 0^{96}|\texttt{0x01}|\texttt{0x00})$. It is noteworthy that unlike the *IRK*, the identity root *IR* is supposed to have 128 bits of entropy according to the standard. In fact, if the *IRK* is all-zero, then the device does not support resolvable private address. We assume in the following that the *IRK* is created randomly and non-zero.

With an *IRK*, the device can generate a (pseudo)random address as follows:

$$\texttt{BD\_ADDR} \leftarrow [prand \mid \mathsf{AES}(IRK, 0^{104}|prand) \bmod 2^{24}],$$

where the 24-bit value *prand* consists of the 22 random bits and '01'. In order to resolve the obtained random private address `BD_ADDR`, the receiving device extracts *prand* out of the received address. Then the device goes through its list of stored *IRK*s and for each entry checks whether the AES-computation with that *IRK* for the (padded) value *prand* matches the `BD_ADDR`. If so, it can look up the actual address of the device and the long-term key, stored together with the *IRK*. If the device does not find a matching *IRK* in the list, then it ignores the PDU from the other party.

Devices achieve privacy only if they have bonded and exchanged the necessary keys, *IRK* and *CSRK*, as well as the identities (either static random addresses or a public addresses). The exchange of these data happens after the devices have performed the initial connection and enabled encryption. The devices generate their keys and exchange it encrypted in the following order. First, the Peripheral sends its *IRK*, address, and *CSRK*. Then the exchange is followed by the Central sending the information in the same order. This means that both parties share their *IRK* with any other bonded device, but the exchange is done over a secured communication channel. The specification also allows IRKs to be pre-distributed. However, we do not consider this case here since it requires assumptions on the channel during the pre-distribution procedure.

Discovery Phase. Link establishment starts with a discovery process. During this process, two devices in proximity synchronize, by one device advertising and the other scanning for potential connections. The link layer Central is called the initiator, and the link layer Peripheral is called the responder. The advertising protocol data unit (PDU) has the following format:

structure	Header						Payload			
field	*PDUtype*	*RFU*	*ChSel*	`TxAdd`	`RxAdd`	*Length*	*AdvA*	AD_1	AD_2	...
bits	4	1	1	1	1	8	48	variable	variable	...

The important for privacy information contained in the packets are the Bluetooth addresses `BD_ADDR` in the *AdvA* field in the payload, which can be one of the four aforementioned types. The flags `TxAdd` and `RxAdd` in the header indicate whether the transmission address (`TxAdd`) resp. reception address (`RxAdd`) is random (= 1) or public (= 0). The Payload may contain additional advertisement data (AD) elements, like the AD type flag and AD data. The latter can be for example a human-readable "complete local name". We simply write $AD_1, AD_2, \ldots$ for these data elements.

The entries *PDUtype* contain the advertisement type, *RFU* is reserved for future use, *ChSel* determines whether the device supports an alternative channel selection algorithm, *Length* describes the length of the payload.

Pairing Feature Extraction. Once the devices have established the link, the pairing starts with the pairing request and response. This information determines the features how the two devices can pair. The pairing requests contain the following information:

field	*Code*	*IOcap*	*OOB*	*AuthReq*						*MaxEnc*	*InitKey*					*RespKey*
sub				*BF*	*MITM*	*SC*	*KP*	*CT2*	*Rsrv*		*LTK*	*IRK*	*CSRK*	*LK*	*Rsrv*	
bits	8	8	8	2	1	1	1	1	2	8	1	1	1	1	4	8

The most relevant for privacy entries here are *SC*: the bit that indicates whether the device supports the "Secure Connections" mode. If both parties have this flag set, then the devices use the P-256 elliptic curve, else they go for the legacy mode. Bit *BF* defines whether two pairing devices will create a bond (i.e., store the security and identity information, such as *LTK*, *IRK* *CSRK*) or not. The other important entry is the *IOcap* byte, which describes the input/output capabilities of the device.

The entry *MaxEnc* sets the number of octets for encryption keys. The lack of authentication of the entries enabled the "Key Negotiation of Bluetooth" (KNOB) attack [ATR20b] where the machine-in-the-middle adversary sets the entry to 7 bytes for long-term keys in BLE, making the devices use a weak key. To prevent this downgrade attack, devices should only support 128-bit keys. We presume that this countermeasure is in place.

The further entries are as follows: The entry *Code* determines whether this is a request or response; *OOB* specifies whether OOB data is available; *BF* says whether the device supports bonding; *MITM* determines whether the device requests to use machine-in-the-middle protection (e.g., if neither *OOB* nor *MITM* are set on the devices, then they revert to JustWorks connections; if the *OOB* flags are not set and at least one device sets *MITM*, then they use *IOcap* to determine the connection method); *KP* is the keypress flag used in the passkey entry mode, *CT2* defines what is used as input to AES-CMAC for generation of an intermediate key when conversing *LTK* to *LK* and the other way around.

The initiator and responder distribution-key entries *InitKey* and *RespKey* contain information used in the optional "Transport Specific Key Distribution" phase that determines the data exchanged when bonding. For Secure Connections, the Central or the Peripheral can later send either of the following information: the "Identity Resolving Key" *IRK* to resolve pseudorandom addresses when reconnecting; the public, or static random address; and the "Connection Signature Resolving Key" *CSRK* to authenticate (unencrypted) data. We stress that the flags here only indicate which keys should be distributed; the actual data is exchanged later.

We note that all these data are sent in clear. This potentially allows distinguishing devices based on their features. This is inevitable, therefore we aim in the following to protect only devices with identical features and focus only on the cryptographic transcript part.

5.2 Privacy Requirements

The Bluetooth protocol aims to hide a device's identity if private address resolution is used and against outsiders with which the private address resolution has not been established [BT523, Vol 3, Part H, Section 2.4.2.1]:

> "The privacy concept only protects against devices that are not part of the set to which the IRK has been given."

Since any communication with the adversary controlling some device would reveal the *IRK*, we thus only consider executions between devices in which the adversary is passive.

To capture this behavior, we give the adversary only a Test oracle, which it can query about three devices. One device serves as the communication partner with one of the other two devices, where the choice is made at random according to some challenge bit b. The devices either start a new initial connection or reconnect, and the adversary gets to learn the transcript of the communication. The task of the adversary is to predict the bit b. To avoid trivial attacks, we assume that two devices in question either both share an *IRK* with the other device or neither of them.

Formally, the Test oracle takes as input three identities $i_0, i_1, j \in \mathcal{I}$ of devices and a value mode, either equal to init or to reconnect, and some auxiliary information aux (e.g., describing the requested SSP protocol). The oracle, holding the random challenge bit b, runs an execution between device i_b and j according to the parameters and returns the transcript to the adversary.

As mentioned before, the distribution of *IRK* and BD_ADDR happens after the devices have enabled encryption. Therefore, we extend the initial connection procedure by forcing the devices to enable encryption and perform the key-distribution step. If this does not happen, the pairing step (and hence the initial connection) fails and the devices are not considered bonded.

To strengthen the definition, we assume that the adversary learns all actual addresses of the devices at the outset. We may for simplicity assume that the identity i of a device equals this address. For initialization, we also assume that a secret key, called *IRK* here as well, is generated at the beginning of the security experiment.

Definition 5.1 (Outsider Privacy) *The Bluetooth protocol* Π *provides outsider privacy if for any PPT adversary* $\mathcal{A}$

$$\boldsymbol{Adv}^{\mathit{Privacy}}_{\mathcal{A},\Pi}(\lambda) := \Pr\Big[\boldsymbol{Exp}^{\mathit{Privacy}}_{\mathcal{A},\Pi}(\lambda) = 1\Big] - \frac{1}{2}$$

is negligible, where

```
$\boldsymbol{Exp}^{\mathit{Privacy}}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda)$
$b \leftarrow_{\$} \{0,1\}$
forall $i \in \mathcal{I}$ do
    $\mathit{IRK} \leftarrow_{\$} \{0,1\}^{128} \setminus \{0\}$
$a \leftarrow_{\$} \mathcal{A}^{\mathsf{Test}}(\mathcal{I})$
return 1 if $a = b$
```

5.3 Privacy Guarantees of BLE

We say that a device running BLE is in *full privacy mode* if it uses a non-resolvable random private address when establishing an initial connection to some other device, and a resolvable one when reconnecting to that device. Furthermore, we assume devices use a fresh Diffie–Hellman value in each SSP execution.

Proposition 5.2 (Outsider Privacy) *The Bluetooth LE protocol* Π_{BLE} *in full privacy mode provides outsider privacy. That is, for any adversary* $\mathcal{A}$ *calling at most* q_s *test sessions, there exists an adversary* $\mathcal{B}$ *(with roughly the same run time as* $\mathcal{A}$*) such that*

$$\boldsymbol{Adv}^{\mathit{Privacy}}_{\mathcal{A},\Pi_{\mathrm{BLE}},\mathcal{I}}(\lambda) \leq q_s^2 \cdot 2^{-|\mathit{prand}|+2} + q_s \cdot \boldsymbol{Adv}^{\mathit{PRF}}_{\mathcal{B},\mathsf{AES}}(\lambda).$$

where $|\mathit{prand}| = 24$*.*

Note that two bits of *prand* are reserved to signal the address type such that *prand* only consists of 22 random bits. We remark that the bound is tight in the sense that there is an adversary that can link a device (and thus predict the challenge bit) with probability $q_s^2 \cdot 2^{-|\mathit{prand}|+2}$. For this, the adversary considers one device (with identity j) and one target device (with identity t) and initializes q_s other devices. It connects each of the $q_s + 1$ devices to j such that they all share an individual *IRK* with device j. Then it calls the Test oracle to reconnect device j to either device t, or to the next unused additional device. If at some point the same random address appears twice then the adversary concludes that the secret bit b is 0 and the target device t is communicating. If no such collision occurs then the attacker outputs a random bit.

For the analysis, note that if the Test oracle always picks the device t with the same *IRK*, i.e., $b = 0$, then a collision on *prand* implies a collision on the full address. Hence, this happens with probability roughly $q_s^2 \cdot 2^{-22}$. For different devices and fresh *IRK*s, this happens rarely, with probability approximately $q_s^2 \cdot 2^{-46}$, even if the *prand* values collide. The difference in probabilities is thus still in the order of $q_s^2 \cdot 2^{-22}$. If neither case occurs, then our attacker succeeds with probability $\frac{1}{2}$ by the random guess, such that the overall advantage is close to $q_s^2 \cdot 2^{-22}$.

Proof (of Proposition 5.2). We proceed once more by a game-hopping argument. We denote again by $\Pr[\mathsf{Game}_j]$ the probability that the adversary wins in the corresponding game (over the guessing probability).

Game 0. Game Game_0 is the original attack on the privacy.

Game 1. We declare the adversary to lose if the *prand* parts of the initially transmitted resolvable addresses in any pair of reconnection calls to Test collide.

Note that since each device chooses 22-bits of the value *prand* randomly, the probability of such a collision, independently of the question whether the test oracle uses the left or right device, is given by at most $q_s^2 \cdot 2^{-22}$. Hence, $\Pr[\mathsf{Game}_0] \leq \Pr[\mathsf{Game}_1] + q_s^2 \cdot 2^{-22}$.

Game 2. In Game_2, we replace the most significant 24 pseudorandom bits in this resolvable private random addresses transmitted or used in a reconnection step by independent random bits (chosen randomly once but fixed in this execution). Internally, the receiving party of such a modified address will be told the correct entry in the list.

Starting with Game_1, we first replace the pseudorandom functions $\mathsf{AES}(\mathit{IRK}, \cdot)$ for each distinct IRK by a random function (but using the same random function for re-appearing IRK's). We can do this by a hybrid argument among the (at most) q_s different keys IRK, simulating the other game steps. Note that we can identify re-appearing IRKs by looking at the identities of devices. This step occurs a loss of $q_s \cdot \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{B},\mathsf{AES}}(\lambda)$, where $\mathcal{B}$ is the game-simulating adversary. We now apply a random function to different inputs, since all *prand* values are distinct by the previous game hop. This effectively means that all the 24-bit outputs are random. This corresponds now exactly to Game_2.

We finally note that all the cryptographic parts in transcripts generated by the Test oracle are independent of the device. In initial connections, the device i_b in a Test query uses a non-resolvable private random address and a fresh Diffie–Hellman value, by the assumption about the full privacy mode of the device. All other protocol steps of an SSP run are neither device-specific. (Note that the addresses used in the protocol are the now updated values, and that we assume that the IO capabilities of the devices i_0, i_1 in a Test query must be equal.)

In each reconnection step, the resolvable private random address is now purely random, and otherwise the parties only exchange random values $\mathit{SKD\_C}, \mathit{IV\_C}$ and $\mathit{SKD\_P}, \mathit{IV\_P}$. It follows that this step does not depend on the device in question. Since each Test oracle query in the final game is therefore independent of any device-specific data, the adversary cannot do better in the final game than guessing the challenge bit b. □

6 Conclusion

Our results complement the long list of successful attacks on the Bluetooth protocol suite. These attacks exploit dependencies between different subprotocols or even between the BR/EDR and BLE technology, or the possibility to downgrade the data. We show that if one sticks to the strongest connection model, then the only attack possibility against key secrecy is to be active during the initial connection step. Otherwise the encryption keys are secret, albeit the role of the parties nor their identity are authenticated.

Based on our experience with the analysis of the Bluetooth standard, we would like to conclude that the standard is hard to digest, both in terms of size as well as in terms of clarity. Especially when it comes to the desired security properties, the standard is rather vague in the sense that the requirements are not specified or subsumed under imprecise terms. To give an example, the term "authentication" is used in several contexts with different meanings. It could be entity authentication in the sense that the devices' identities are confirmed, or key authentication in the sense that only intended partner derive the session key, or a form of protection against machine-in-the-middle attacks. The Authentication stage 2 in the SSP protocol rather seems to be a key confirmation step.

Acknowledgments

We thank the anonymous reviewers for extremely comprehensive and helpful comments. This research work has been funded by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. It has also been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) — 251805230/GRK 2050.

References

[Ant23] Daniele Antonioli. BLUFFS: Bluetooth forward and future secrecy attacks and defenses. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023: 30th Conference on Computer and Communications Security*, pages 636–650, Copenhagen, Denmark, November 26–30, 2023. ACM Press. (Cited on page 5.)

[ATR19] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Bonne Rasmussen. The KNOB is broken: Exploiting low entropy in the encryption key negotiation of bluetooth BR/EDR. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019: 28th USENIX Security Symposium*, pages 1047–1061, Santa Clara, CA, USA, August 14–16, 2019. USENIX Association. (Cited on pages 5 and 13.)

[ATR20a] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. BIAS: Bluetooth impersonation AttackS. In *2020 IEEE Symposium on Security and Privacy*, pages 549–562, San Francisco, CA, USA, May 18–21, 2020. IEEE Computer Society Press. (Cited on pages 5 and 6.)

[ATR20b] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. Key negotiation downgrade attacks on bluetooth and bluetooth low energy. *ACM Trans. Priv. Secur.*, 23(3):14:1–14:28, 2020. (Cited on pages 5, 13, and 25.)

[ATRP20] Daniele Antonioli, Nils Ole Tippenhauer, Kasper Rasmussen, and Mathias Payer. BLURtooth: Exploiting Cross-Transport Key Derivation in Bluetooth Classic and Bluetooth Low Energy. 2020. (Cited on pages 5 and 6.)

[BFGJ17] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. PRF-ODH: Relations, instantiations, and impossibility results. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 651–681, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland. (Cited on page 18.)

[BLS19] Johannes K. Becker, David Li, and David Starobinski. Tracking anonymized bluetooth devices. *Proceedings on Privacy Enhancing Technologies*, 2019(3):50–65, July 2019. (Cited on page 8.)

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984. (Cited on page 18.)

[BN19] Eli Biham and Lior Neumann. Breaking the bluetooth pairing - the fixed coordinate invalid curve attack. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019: 26th Annual International Workshop on Selected Areas in Cryptography*, volume 11959 of *Lecture Notes in Computer Science*, pages 250–273, Waterloo, ON, Canada, August 12–16, 2019. Springer, Cham, Switzerland. (Cited on pages 7, 11, and 19.)

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Berlin, Heidelberg, Germany. (Cited on page 14.)

[BT521] Bluetooth core specification. Bluetooth Special Interest Group (SIG), July 2021. Ver. 5.3. (Cited on page 3.)

[BT523] Bluetooth core specification. Bluetooth Special Interest Group (SIG), January 2023. Ver. 5.4. (Cited on pages 3, 7, 8, 11, 18, 23, 25, 33, 35, 39, and 42.)

[CADLE23] Tristan Claverie, Gildas Avoine, Stéphanie Delaune, and José Lopes-Esteves. Tamarin-based analysis of bluetooth uncovers two practical pairing confusion attacks. In Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis, editors, *ESORICS 2023: 28th European Symposium on Research in Computer Security, Part III*, volume 14346 of *Lecture Notes in Computer Science*, pages 100–119, The Hague, The Netherlands, September 25–29, 2023. Springer, Cham, Switzerland. (Cited on page 4.)

[CC19a] Guillaume Celosia and Mathieu Cunche. Fingerprinting bluetooth-low-energy devices based on the generic attribute profile. In Peng Liu and Yuqing Zhang, editors, *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things, IoT S&P@CCS 2019, London, UK, November 15, 2019*, pages 24–31. ACM, 2019. (Cited on page 8.)

[CC19b] Guillaume Celosia and Mathieu Cunche. Saving private addresses: an analysis of privacy issues in the bluetooth-low-energy advertising mechanism. In H. Vincent Poor, Zhu Han, Dario Pompili, Zhi Sun, and Miao Pan, editors, *MobiQuitous 2019, Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Houston, Texas, USA, November 12-14, 2019*, pages 444–453. ACM, 2019. (Cited on page 8.)

[CCG$^+$19] Katriel Cohn-Gordon, Cas Cremers, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. Highly efficient key exchange protocols with optimal tightness. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 767–797, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. (Cited on page 20.)

[CGP$^+$20] Marco Cominelli, Francesco Gringoli, Paul Patras, Margus Lind, and Guevara Noubir. Even black cats cannot stay hidden in the dark: Full-band de-anonymization of bluetooth classic devices. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 534–548. IEEE, 2020. (Cited on page 7.)

[CL21] Tristan Claverie and José Lopes-Esteves. Bluemirror: Reflections on bluetooth pairing and provisioning protocols. In *IEEE Security and Privacy Workshops, SP Workshops 2021, San Francisco, CA, USA, May 27, 2021*, pages 339–351. IEEE, 2021. (Cited on page 4.)

[DJ21] Denis Diemert and Tibor Jager. On the tight security of TLS 1.3: Theoretically sound cryptographic parameters for real-world deployments. *Journal of Cryptology*, 34(3):30, July 2021. (Cited on page 20.)

[GJ18] Kristian Gjøsteen and Tibor Jager. Practical and tightly-secure digital signatures and authenticated key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in*

Cryptology – CRYPTO 2018, Part II, volume 10992 of *Lecture Notes in Computer Science*, pages 95–125, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. (Cited on page 20.)

[HN04] Johan Håstad and Mats Näslund. The security of all RSA and discrete log bits. *J. ACM*, 51(2):187–230, 2004. (Cited on page 18.)

[JKSS12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 273–293, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Heidelberg, Germany. (Cited on page 6.)

[JVAF21] Loïc Jouans, Aline Carneiro Viana, Nadjib Achir, and Anne Fladenmuller. Associating the randomized bluetooth MAC addresses of a device. In *18th IEEE Annual Consumer Communications & Networking Conference, CCNC 2021, Las Vegas, NV, USA, January 9-12, 2021*, pages 1–6. IEEE, 2021. (Cited on page 8.)

[Lin09] Andrew Y. Lindell. Comparison-based key exchange and the security of the numeric comparison mode in Bluetooth v2.1. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 66–83, San Francisco, CA, USA, April 20–24, 2009. Springer, Berlin, Heidelberg, Germany. (Cited on pages 6 and 7.)

[LVHNN21] Norbert Ludant, Tien D. Vo-Huu, Sashank Narain, and Guevara Noubir. Linking bluetooth le & classic and implications for privacy-preserving bluetooth-based protocols. In *2021 IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, May 24-27, 2021.* IEEE, 2021. (Cited on page 8.)

[SS19] Da-Zhi Sun and Li Sun. On secure simple pairing in bluetooth standard v5.0-part i: Authenticated link key security and its home automation and entertainment applications. *Sensors*, 19(5), 2019. (Cited on pages 6 and 7.)

[SSY19] Da-Zhi Sun, Li Sun, and Ying Yang. On secure simple pairing in bluetooth standard v5.0-part II: privacy analysis and enhancement for low energy. *Sensors*, 19(15):3259, 2019. (Cited on page 7.)

[TH21] Michael Troncoso and Britta Hale. The bluetooth CYBORG: Analysis of the full human-machine passkey entry AKE protocol. In *ISOC Network and Distributed System Security Symposium – NDSS 2021*, Virtual, February 21–25, 2021. The Internet Society. (Cited on pages 4, 6, 7, and 20.)

[VMC$^+$16] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S. Cardoso, and Frank Piessens. Why MAC address randomization is not enough: An analysis of wi-fi network discovery mechanisms. In Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang, editors, *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016*, pages 413–424. ACM, 2016. (Cited on page 8.)

[vTPFG21] Maximilian von Tschirschnitz, Ludwig Peuckert, Fabian Franzen, and Jens Grossklags. Method confusion attack on bluetooth pairing. In *2021 IEEE Symposium on Security and Privacy*, pages 1332–1347, San Francisco, CA, USA, May 24–27, 2021. IEEE Computer Society Press. (Cited on pages 4, 5, and 6.)

[WNK$^+$20] Jianliang Wu, Yuhong Nan, Vireshwar Kumar, Dave (Jing) Tian, Antonio Bianchi, Mathias Payer, and Dongyan Xu. BLESA: Spoofing attacks against reconnections in bluetooth low energy. In *14th USENIX Workshop on Offensive Technologies (WOOT 20)*. USENIX Association, August 2020. (Cited on page 5.)

[ZWD$^+$20] Yue Zhang, Jian Weng, Rajib Dey, Yier Jin, Zhiqiang Lin, and Xinwen Fu. Breaking secure pairing of bluetooth low energy using downgrade attacks. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020: 29th USENIX Security Symposium*, pages 37–54. USENIX Association, August 12–14, 2020. (Cited on page 4.)

A Other Subprotocols for Association Models in SSP

In this section, we describe the other subprotocols for the different association models used in the SSP protocol: JustWorks, PasskeyEntry, and OOB. The execution of NumCom subprotocol is shown in Figure 4 in Section 2.2. We note that only the Authentication stage 1 in SSP differs for the different models and the other stages are identical for all association models (apart from OOB wherein additional step with pre-distribution of the information is needed, see A.3 for details).

The models are chosen based on the IO capabilities of the devices according to Table 1.

A.1 Just Works

The Authentication stage 1 in the association model JustWorks does not perform any checks involving the user. It can be seen as a simplified version of NumCom without the user checking the 6-digit display. The check and confirmation of the commitment is done hence automatically by the initiating device. The protocol is displayed in Figure 8.

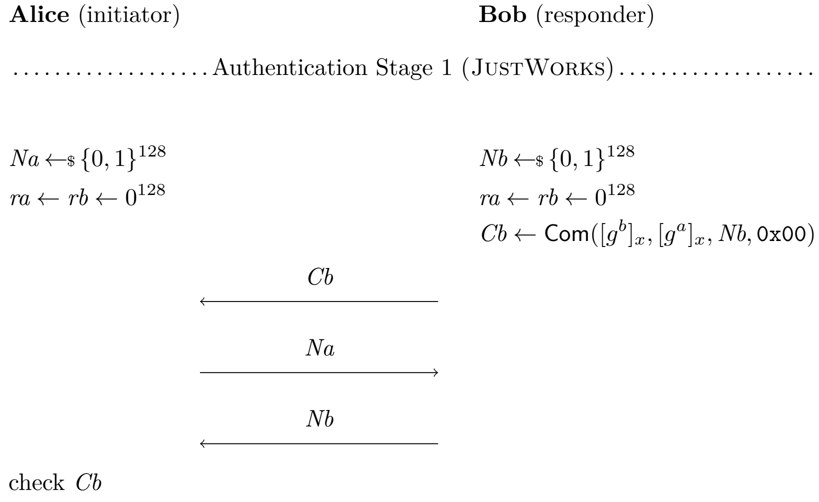
Alice (initiator)
Bob (responder)
Authentication Stage 1 (JustWorks)
$Na \leftarrow_{\$} \{0,1\}^{128}$
$ra \leftarrow rb \leftarrow 0^{128}$
$Nb \leftarrow_{\$} \{0,1\}^{128}$
$ra \leftarrow rb \leftarrow 0^{128}$
$Cb \leftarrow \mathsf{Com}([g^b]_x, [g^a]_x, Nb, \mathtt{0x00})$
Cb
Na
Nb
check Cb

Figure 8: Bluetooth Authentication stage 1 of the Secure Simple Pairing protocol in mode Just Works.

A.2 Passkey Entry

The Authentication stage 1 of the PasskeyEntry association model requires the user enter (on both or just one side) a 6-digit passkey. In the case of one-side entering, one device displays the digits that need to be entered by a user on the other device. We define this by distribution of the passkey happening outside of the protocol, i.e., the devices simply inherit it, even if one device generates it randomly in order to display to a user.

This passkey value is transformed into a 20-bit string and then both parties run 20 check rounds, each with one bit (padded with `0b1000000` in each round since the packet must contain at least one byte) of the 20-bit passkey value. In each round, the parties pick fresh 128-bit nonces. The nonces in the final rounds are set to be *Na* and *Nb*; and the passkey is padded with 0 to form a 128-bit long string. The protocol is displayed in Figure 9.

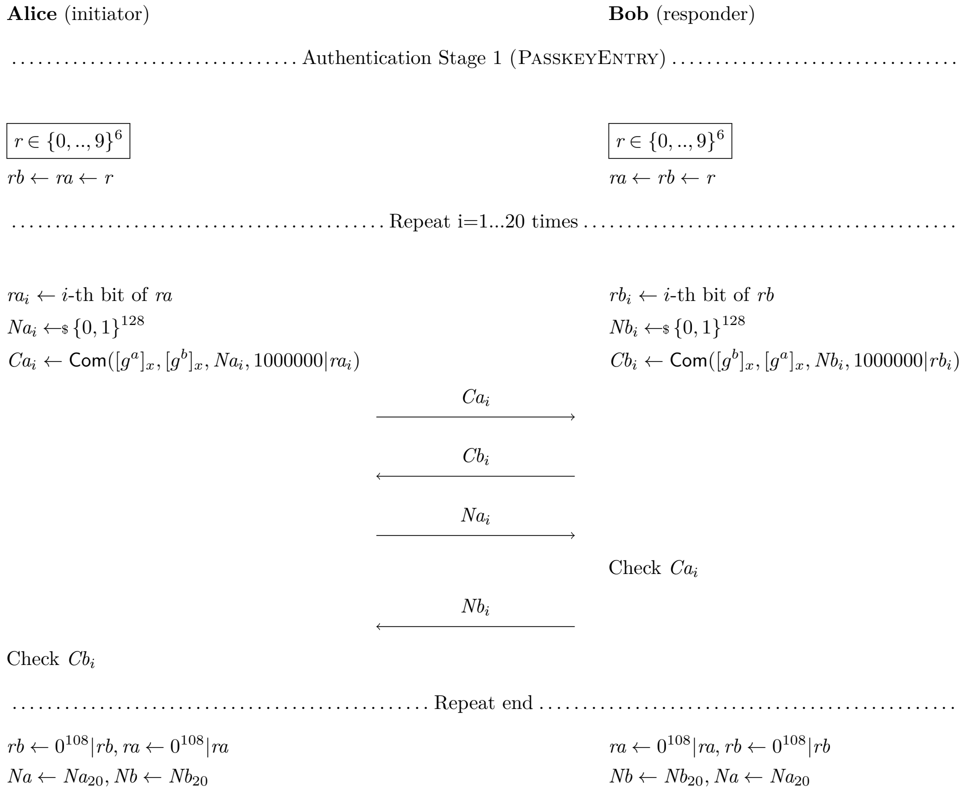
Alice (initiator)
Bob (responder)
Authentication Stage 1 (PasskeyEntry)
$\boxed{r \in \{0, .., 9\}^6}$
$rb \leftarrow ra \leftarrow r$
$\boxed{r \in \{0, .., 9\}^6}$
$ra \leftarrow rb \leftarrow r$
Repeat i=1...20 times
$ra_i \leftarrow i$-th bit of ra
$Na_i \leftarrow_{\$} \{0,1\}^{128}$
$Ca_i \leftarrow \mathsf{Com}([g^a]_x, [g^b]_x, Na_i, 1000000|ra_i)$
$rb_i \leftarrow i$-th bit of rb
$Nb_i \leftarrow_{\$} \{0,1\}^{128}$
$Cb_i \leftarrow \mathsf{Com}([g^b]_x, [g^a]_x, Nb_i, 1000000|rb_i)$
$\xrightarrow{Ca_i}$
$\xleftarrow{Cb_i}$
$\xrightarrow{Na_i}$
Check Ca_i
$\xleftarrow{Nb_i}$
Check Cb_i
Repeat end
$rb \leftarrow 0^{108}|rb, ra \leftarrow 0^{108}|ra$
$Na \leftarrow Na_{20}, Nb \leftarrow Nb_{20}$
$ra \leftarrow 0^{108}|ra, rb \leftarrow 0^{108}|rb$
$Nb \leftarrow Nb_{20}, Na \leftarrow Na_{20}$

Figure 9: Bluetooth Authentication stage 1 in the Secure Simple Pairing protocol in mode Passkey Entry.

A.3 Out-of-Band

In OOB association model, the devices exchange some data via an alternative (out-of-band) channel. The data includes the Bluetooth address of a device, commitments of the public keys, and random values that are used in further execution. The out-of-band communication precedes the discovery of the devices and protocol execution. The reversed order, when OOB communication happens during the protocol execution, is not supported by Bluetooth [BT523, Vol 1, Part A, Section 5.2.4.3]. The Authentication stage 1 of the OOB association model as well as OOB communication are displayed in Figure 10.

During the discovery and feature communication phase, the devices check whether they have received the values from the other device throughout OOB communication and still have the information present. If this is true, they set the OOB flag correspondingly. We note that the protocol allows the case of only one device being able to transmit data via another channel. However, even in this case both devices exchange the nonces during the Authentication stage 1.

If OOB pairing failed, the devices choose the other pairing method depending on their IO capabilities and proceed immediately.

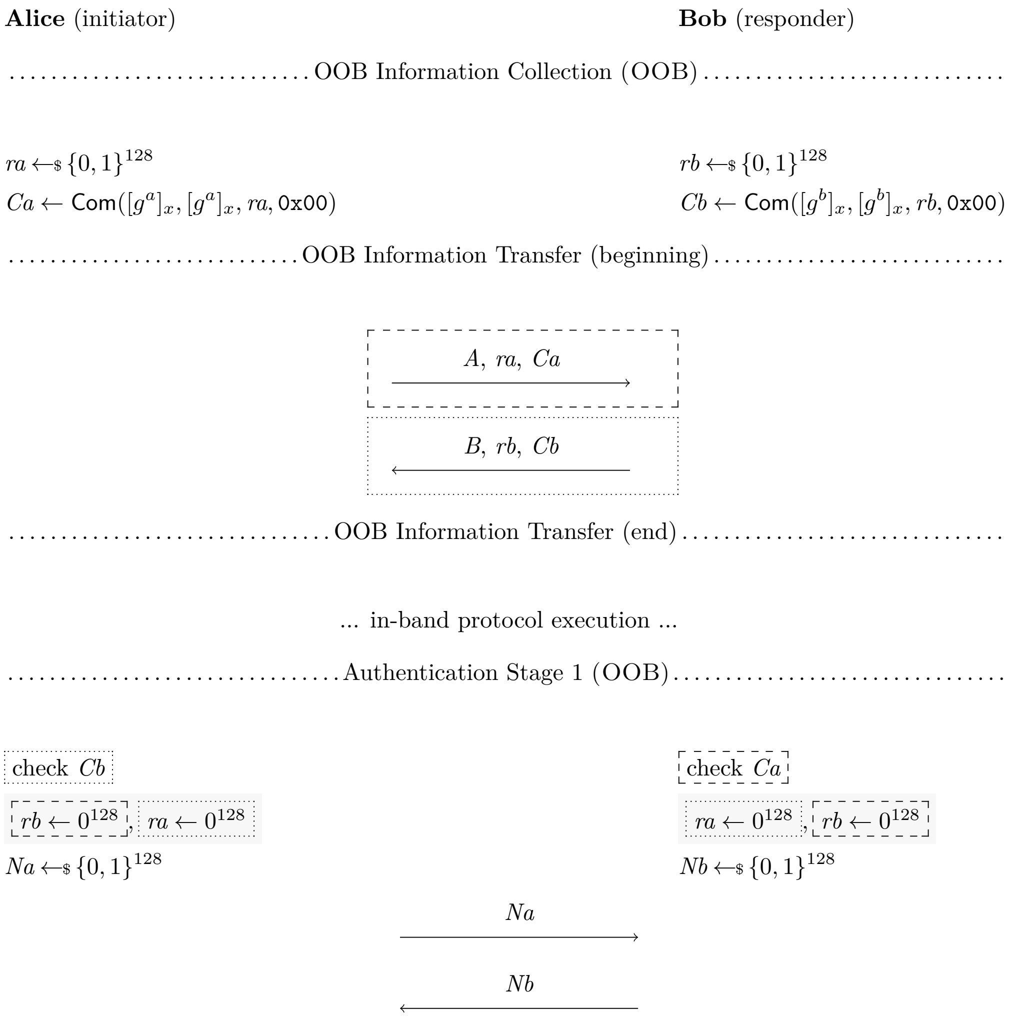
Alice (initiator)
Bob (responder)
OOB Information Collection (OOB)
$ra \leftarrow_{\$} \{0,1\}^{128}$
$Ca \leftarrow \mathsf{Com}([g^a]_x, [g^a]_x, ra, \mathtt{0x00})$
$rb \leftarrow_{\$} \{0,1\}^{128}$
$Cb \leftarrow \mathsf{Com}([g^b]_x, [g^b]_x, rb, \mathtt{0x00})$
OOB Information Transfer (beginning)
$A,\ ra,\ Ca$
$B,\ rb,\ Cb$
OOB Information Transfer (end)
... in-band protocol execution ...
Authentication Stage 1 (OOB)
check Cb
$rb \leftarrow 0^{128}$, $ra \leftarrow 0^{128}$
$Na \leftarrow_{\$} \{0,1\}^{128}$
check Ca
$ra \leftarrow 0^{128}$, $rb \leftarrow 0^{128}$
$Nb \leftarrow_{\$} \{0,1\}^{128}$
Na
Nb

Figure 10: Bluetooth Authentication stage 1 in Secure Simple Pairing in mode Out-of-Band. Gray color indicates the steps that are used in one-way data transmission only. Dashed steps stand for the case of A transmitting data resp. dotted for B transmitting data.

B Acronyms, Glossary, and Variables

In this section, we give three lists for easier navigation throughout the paper. The first list in B.1 consists of acronyms that are used in the paper and the Core Specification [BT523]. The second list in B.2 contains the terms from the Core Specification adapted to the paper. The last list in B.3 includes variables used in the paper and the Core Specification.

B.1 Acronyms

BLE	Bluetooth Low Energy
BR/EDR	Basic Rate/Enhanced Data Rate
CSRK	Connection Signature Resolving Key
DDH	Decisional Diffie–Hellman
ECDH	Elliptic-Curve Diffie–Hellman
GATT	Generic attribute profile
IO	Input/Output
IRK	Identity Resolving Key
IV	Initialization vector
LK	Link Key
LTK	Long-term key
L(T)K	Long-term and Link key
MAC	Media Access Control *or* Message Authentication Code
MITM	Monster in the Middle
ODH	Oracle Diffie–Hellman
OOB	Out of Band
PDU	Protocol Data Unit
PPT	Probabilistic Polynomial-time
PRF	Pseudorandom Function
PRF-ODH	Pseudorandom-function oracle-Diffie–Hellman
SSP	Secure Simple Pairing
SC	Secure Connections
TOFU	Trust on the first use
UUID	Universally Unique Identifier

B.2 Glossary

A

Association model is one of the four subprotocols (JustWorks, NumCom, OOB, PasskeyEntry) in the SSP protocol for nonce exchange. See more in [Vol 1, Part A, Section 5.2.4].

Authentication is a challenge-response procedure to authenticate bonded devices in BR/EDR. See more in Section 2.3 or in [Vol 2, Part H, Section 5].

B

Bonded devices are the devices that had successful initial connection.

Bonding is an optional procedure of storing the connection key established during the pairing process. Bonding is necessary to form long-term relationship between two devices. See more in [Vol 1, Part

A, Section 5.1].

C

Connection key is a secret key derived during pairing. It corresponds to LK for BR/EDR and LTK for BLE (SC). The connection key can be stored via bonding procedure for future reconnections.

Connection Signature Resolving Key (*CSRK*) is a 128-bit key used to sign data and verify signatures on the receiving device [Vol 3, Part H, Section 2.4.1]. See more in [Vol 3, Part H, Section 2.4.2.2].

D

Dedicated Bonding is a type of bonding aiming for creation of the bond between two devices without accessing any service. See more in [Vol 3, Part C, Section 6.5.3.2].

E

Encryption key ($\mathsf{k}_{\mathbf{AES}}$) is an symmetric encryption key generated during the Session Key Agreement procedure.

Encryption Key Generation is a procedure to generate an encryption key (AES key) $\mathsf{k}_{\mathrm{AES}}$ in BR/EDR. See more in Section 2.3 or in [Vol 2, Part H, Section 3.2.5].

G

General Bonding is a type of bonding that is necessary done for consequential service accessing. See more in [Vol 3, Part C, Section 6.5.3.1].

I

Identity Resolving Key (*IRK*) is a 128-bit key used to generate and resolve random addresses [Vol 3, Part H, Section 2.4.1]. See more in Section 5.1 or in [Vol 3, Part H, Section 2.4.2].

Initial connection is a set of actions (radio link establishment, pairing, and bonding), necessary to establish the bond between two devices.

L

Link Key (*LK*) is a connection key in BR/EDR.

Link Layer Encryption is a procedure to generate an encryption key (AES key) $\mathsf{k}_{\mathrm{AES}}$ in BLE. See more in Section 2.3 and in [Vol 6, Part B, Section 5.1.3].

Long-term key (*LTK*) is a connection key in BLE SC.

P

Pairing is the process of establishing a connection key between two devices: LK for BR/EDR and LTK for BLE (SC). In BLE Legacy, STK is used instead, which might be employed for LTK distribution. See more in [Vol 1, Part A, Section 5.1].

R

Radio link establishment is a set of actions necessary to create and maintain a physical link between the devices. We leave the details out of the scope of the paper and refer to radio link establishment mainly as inquiry, page, and IO capabilities exchange for BR/EDR; and as discovery and pairing-feature extraction for BLE.

Reconnection is a process of establishing the subsequent connection between two bonded devices. Reconnection consists of radio link establishment or/and Session Key Agreement.

S

Session Key Agreement is a procedure to establish a shared encryption key to encrypt the communication. In BD/EDR, it consists of Authentication and Encryption Key Generation. In BLE, it consists of Link Layer Encryption.

B.3 Variables

Variable	Name	Size	Value/Origin	Used in	References
A	Identity of device A	-	No value or set to BD_ADDR$_A$	Device identification	-
ACO	Authentication Ciphering Offset	64 bits	65-128 most significant bits from HMAC(dk,$AU\_RAND\_C|AU\_RAND\_P$)	$\mathsf{k}_{\mathrm{AES}}$ generation in BR/EDR	Section 2.3 [Vol 2, Part H, Section 5]
AU_RAND_C	Central's Authentication Random Number	128 bits	Randomly generated by a device (if Central) or received from the other device (if Peripheral)	As Random input (challenge) in BR/EDR: $SRES\_C$, $SRES\_P$, and ACO generation	Section 2.3 [Vol 2, Part H, Section 3.2.2]
AU_RAND_P	Peripheral's Authentication Random Number	128 bits	Randomly generated by a device (if Peripheral) or received from the other device (if Central)	As Random input (challenge) in BR/EDR: $SRES\_C$, $SRES\_P$, and ACO generation	Section 2.3 [Vol 2, Part H, Section 3.2.2]
$AuthReq$	Authentication Requirements flags	1 byte	BR/EDR: 0x00: No MITM Protection Required – No Bonding 0x01: MITM Protection Required – No Bonding 0x02: No MITM Protection Required – Dedicated Bonding 0x03: MITM Protection Required – Dedicated Bonding 0x04: No MITM Protection Required – General Bonding 0x05: MITM Protection Required – General Bonding BLE: concatenation of BF, $MITM$, SC, KP, $CT2$, and 0b00	Determination of the association model for pairing	Section 5.1 [Vol 2, Part C, Section 5.2] [Vol 3, Part H, Section 3.5.1]
B	Identity of device B	-	No value or set to BD_ADDR$_B$	Device identification	-
BD_ADDR	Bluetooth Device Address	48 bits	BR/EDR: manually assigned MAC address BLE: Public: manually assigned MAC address Static: 0b11 concatenated with 46 randomly generated bits Resolvable private: $prand$ | $\mathsf{AES}(IRK, 0^{104}|prand) \bmod 2^{24}$ Non-resolvable private: 0b00 concatenated with 46 randomly generated bits	Device identification	Section 5.1 [Vol 6, Part B, Section 1.3]
BD_ADDR$_A$	Bluetooth Device Address	48 bits	Known by a device (if A) or received during radio link establishment (if B)	Indication of BD_ADDR of device A	Section 5.1 [Vol 6, Part B, Section 1.3]
BD_ADDR$_B$	Bluetooth Device Address	48 bits	Known by a device (if B) or received during radio link establishment (if A)	Indication of BD_ADDR of device B	Section 5.1 [Vol 6, Part B, Section 1.3]

Variable	Name	Size	Value/Origin	Used in	References
BF	Bonding flags	2 bits	`0b00`: no bonding `0b01`: bonding	As a part of *AuthReq* flag in BLE	Section 5.1 [Vol 3, Part H, Section 3.5.1]
’btak’	String "Bluetooth AES Key"	32 bits	Fixed value `0x6274616B`	As kID_{AES} in k_{AES} computation	Table 3 [Vol 2, Part H, Section 7.7.6]
’btdk’	String "Bluetooth Device Key"	32 bits	Fixed value `0x6274646B`	As kID_{Dev} in dk computation in BR/EDR	Table 3 [Vol 2, Part H, Section 7.7.7]
’btle’	String "Bluetooth Low Energy"	32 bits	Fixed value `0x62746C65`	As kID_{BLE} in *mk* and *LTK* computation	Table 2 [Vol 3, Part H, Section 2.2.7]
’btlk’	String "Bluetooth Link Key"	32 bits	Fixed value `0x62746C6B`	As $kID_{\text{BR/EDR}}$ in *LK* computation	Table 2 [Vol 2, Part H, Section 7.7.3]
Ca	Commitment value from device *A*	128 bits	BR/EDR: 128 most significant bits from $\mathsf{HMAC}([g^a]_x, [g^b]_x, Na, ra)$ BLE: $\mathsf{CMAC}([g^a]_x, [g^b]_x, Na, ra)$	Commitment of *Na*	[Vol 2, Part H, Section 7.7.1] [Vol 3, Part H, Section 2.2.6]
Cb	Commitment value from device *B*	128 bits	BR/EDR: 128 most significant bits from $\mathsf{HMAC}([g^b]_x, [g^a]_x, Nb, rb)$ BLE: $\mathsf{CMAC}([g^b]_x, [g^a]_x, Nb, rb)$	Commitment of *Nb*	[Vol 2, Part H, Section 7.7.1] [Vol 3, Part H, Section 2.2.6]
CT2	*CT2* flag	1 bit	`0`: conversion function h7 is not supported `1`: conversion function h7 is supported	As a part of *AuthReq* flag in BLE: indicates what input will be used for generation of an intermediate key	Section 5.1 [Vol 3, Part H, Section 3.5.1]
dk	Device Key In [BT523]: also device authentication key	128 bits	128 most significant bits from $\mathsf{HMAC}(LK,\text{'btdk'} \mid \texttt{BD\_ADDR}_A \mid \texttt{BD\_ADDR}_B)$	As key in response generation in BR/EDR Authentication procedure	Table 3 [Vol 2, Part C, Section 4.2.1]
Ea	Check value from device *A* In [BT523]: also confirmation value	128 bits	BR/EDR: 128 most significant bits from $\mathsf{HMAC}([g^{ab}]_x, Na, Nb, rb, AuthReqA, OOBa, IOcapA, \texttt{BD\_ADDR}_A, \texttt{BD\_ADDR}_B)$ BLE: $\mathsf{MAC}(mk, Na, Nb, rb, AuthReqA, OOBa, IOcapA, 0^7 \mid \texttt{TxAddA}, \texttt{BD\_ADDR}_A, 0^7 \mid \texttt{TxAddB}, \texttt{BD\_ADDR}_B)$	Confirmation of the shared DHKey	Section 2.2 [Vol 2, Part H, Section 7.3] [Vol 2, Part H, Section 7.7.4] [Vol 3, Part H, Section 2.2.8]
Eb	Check value from device *B* In [BT523]: also confirmation value	128 bits	BR/EDR: 128 most significant bits from $\mathsf{HMAC}([g^{ab}]_x, Nb, Na, ra, AuthReqB, OOBb, IOcapB, \texttt{BD\_ADDR}_B, \texttt{BD\_ADDR}_A)$ BLE: $\mathsf{MAC}(mk, Nb, Na, ra, AuthReqB, OOBb, IOcapB, 0^7 \mid \texttt{TxAddB}, \texttt{BD\_ADDR}_B, 0^7 \mid \texttt{TxAddA}, \texttt{BD\_ADDR}_A)$	Confirmation of the shared DHKey	Section 2.2 [Vol 2, Part H, Section 7.3] [Vol 2, Part H, Section 7.7.4] [Vol 3, Part H, Section 2.2.8]

IOcap	Input/Output Capabilities	8 bits	`0x00` - `DisplayOnly`
			`0x01` - `DisplayYesNo`
			`0x02` - `KeyboardOnly`
			`0x03` - `NoInputNoOutput`
			`0x04` - `KeyboardDisplay` (BLE only)
				Selection of the association model for pairing	Section 2.1
					[Vol 3, Part C, Section 5.2.2.4]
					[Vol 3, Part H, Section 2.3.2]

IOcapA	Input/Output Capabilities of device *A*	8 bits	Known by a device (if *A*) or received upon exchange of device-specific information (if *B*)	Indication of IO capabilities of device *A*	Section 2.1 [Vol 3, Part C, Section 5.2.2.4] [Vol 3, Part H, Section 2.3.2]

Variable	Name	Size	Value/Origin	Used in	References
IOcapB	Input/Output Capabilities of device *B*	8 bits	Known by a device (if *B*) or received upon exchange of device-specific information (if *A*)	Indication of IO capabilities of device *B*	Section 2.1 [Vol 3, Part C, Section 5.2.2.4] [Vol 3, Part H, Section 2.3.2]
IR	Identity Root	128 bits	Randomly generated	Generation of *IRK*	Section 5.1 [Vol 3, Part H, Appendix B.2]
IRK	Identity Resolving Key	128 bits	Derived from *IR* as: $\mathsf{AES}(IR, 0^{96} \vert \texttt{0x0001} \vert \texttt{0x0000})$ or randomly generated	Generation and resolution of Resolvable Private Addresses $\texttt{BD\_ADDR}$	Section 5.1 [Vol 3, Part H, Section 2.4.2] [Vol 3, Part H, Appendix B.2.3]
IRKc	Central's Identity Resolving Key	128 bits	Known by a device (if Central) or received from the other device during Key Distribution (if Peripheral)	Generation and resolution of Resolvable Private Addresses $\texttt{BD\_ADDR}_A$	Section 5.1 [Vol 3, Part H, Section 2.4.2] [Vol 3, Part H, Appendix B.2.3]
IRKp	Peripheral's Identity Resolving Key	128 bits	Known by a device (if Peripheral) or received from the other device during Key Distribution (if Central)	Generation and resolution of Resolvable Private Addresses $\texttt{BD\_ADDR}_B$	Section 5.1 [Vol 3, Part H, Section 2.4.2] [Vol 3, Part H, Appendix B.2.3]
IV	Initialization vector	64 bits	BR/EDR: ACO BLE: $IV\_C \vert IV\_P$	As initialization vector in encryption	Section 2.3 [Vol 2, Part H, Section 9.1] [Vol 6, Part B, Section 5.1.3.1]
IV_C	Central's part of the initialization vector	32 bits	Randomly generated by a device (if Central) or received from the other device (if Peripheral)	As a part of IV in BLE	Section 2.3 [Vol 6, Part B, Section 5.1.3.1]
IV_P	Peripheral's part of the initialization vector	32 bits	Randomly generated by a device (if Peripheral) or received from the other device (if Central)	As a part of IV in BLE	Section 2.3 [Vol 6, Part B, Section 5.1.3.1]
$\mathsf{k}_{\mathrm{AES}}$	AES key	128 bits	BR/EDR: 128 most significant bits from $\mathsf{HMAC}(LK, \mathrm{kID}_{\mathrm{AES}} \vert \texttt{BD\_ADDR}_A \vert \texttt{BD\_ADDR}_B \vert \mathrm{ACO})$ BLE: AES(LTK, $SKD\_C \vert SKD\_P$)	As encryption key for message encryption	Section 2.3 [Vol 2, Part H, Section 7.7.6] [Vol 6, Part B, Section 5.1.3.1]
KP	Keypress flags	2 bits	`0b00`: no notifications `0b01`: generation and sending of SMP Pairing Keypress Notification PDUs	As a part of *AuthReq* flag in BLE	Section 5.1 [Vol 3, Part H, Section 3.5.1]
LK	Link key	128 bits	128 most significant bits from $\mathsf{HMAC}([g^{ab}]_x, Na, Nb, \text{'btlk'}, \texttt{BD\_ADDR}_A, \texttt{BD\_ADDR}_B)$	$\mathsf{k}_{\mathrm{AES}}$ computation and authentication in BR/EDR	Section 2.1 [Vol 2, Part H, Section 7.7.3]
LTK	Long-term key	128 bits	128 least significant bits from $\mathsf{CMAC}(\mathsf{CMAC}(\mathrm{Salt}, [g^{ab}]_x), \texttt{0x01}, \text{'btle'}, Na, Nb, 0^7 \vert \texttt{TxAddA}, \texttt{BD\_ADDR}_A, 0^7 \vert \texttt{TxAddB}, \texttt{BD\_ADDR}_B, \texttt{0x0100})$	$\mathsf{k}_{\mathrm{AES}}$ computation in BLE	Section 2.1 [Vol 3, Part H, Section 2.2.7]
MITM	*MITM* flag	1 bit	`0`: no MITM protection required (JustWorks) `1`: MITM protection required (NumCom, OOB, PasskeyEntry)	As a part of *AuthReq* flag in BLE	Section 5.1 [Vol 3, Part H, Section 3.5.1]
mk	MacKey	128 bits	BR/EDR: $[g^{ab}]_x$ BLE: 128 most significant bits from $\mathsf{CMAC}(\mathsf{CMAC}(\mathrm{Salt}, [g^{ab}]_x), \texttt{0x00}, \text{'btle'}, Na, Nb, 0^7 \vert \texttt{TxAddA}, \texttt{BD\_ADDR}_A, 0^7 \vert \texttt{TxAddB}, \texttt{BD\_ADDR}_B, \texttt{0x0100})$	$\mathsf{k}_{\mathrm{AES}}$ computation in BLE	Section 2.2 [Vol 3, Part H, Section 2.2.7]

Variable	Name	Size	Value/Origin	Used in	References
Na	Nonce (unique random value) from device A	128 bits	Randomly generated by a device (if Central) or received from the other device (if Peripheral)	Variety of the input to LTK and LK KDF	Section 2.2 [Vol 2, Part H, Section 7.2]
Nb	Nonce (unique random value) from device B	128 bits	Randomly generated by a device (if Peripheral) or received from the other device (if Central)	Variety of the input to LTK and LK KDF	Section 2.2 [Vol 2, Part H, Section 7.2]
OOB	OOB Authentication Data Flag	1 byte	0x00: No OOB Authentication Data received 0x01: OOB Authentication Data received	Determination of the association model for pairing	[Vol 2, Part C, Section 5.2] [Vol 3, Part H, Section 3.5.1]
$OOBa$	OOB Authentication Data Flag of device A	1 byte	Known by a device (if A) or received in pairing packets (if B)	Indication whether A has OOB data from B present	[Vol 2, Part C, Section 5.2] [Vol 3, Part H, Section 3.5.1]
$OOBb$	OOB Authentication Data Flag of device B	1 byte	Known by a device (if B) or received in pairing packets (if A)	Indication whether B has OOB data from A present	[Vol 2, Part C, Section 5.2] [Vol 3, Part H, Section 3.5.1]
$prand$	Random part of Resolvable Private Address	24 bit	0b01 concatenated with 22 randomly generated bits	As part of Resolvable Private Address BD_ADDR in BLE	Section 5.1 [Vol 6, Part B, Section 1.3.2.1]
RxAdd	Reception Address Type	1 bit	0 if BD_ADDR is public, 1 if BD_ADDR is random	Indication of the receiver's address type in BLE	Section 5.1 [Vol 6, Part B, Section 2.3]
Salt	Salt	128 bits	Fixed value 0x6C88 8391 AAF5 A538 6037 0BDB 5A60 83BE	Computation of mk and LTK in BLE	Section 2.2 [Vol 3, Part H, Section 2.2.7]
SC	Secure Connections flag	1 bit	0: BLE SC is not supported 1: BLE SC is supported	As a part of $AuthReq$ flag in BLE	Section 5.1 [Vol 3, Part H, Section 3.5.1]
SKD_C	Central's Session key diversifier	64 bits	Randomly generated by a device (if Central) or received from the other device (if Peripheral)	$\mathsf{k}_{\mathrm{AES}}$ generation in BLE	Section 2.3 [Vol 6, Part B, Section 5.1.3.1]
SKD_P	Peripheral's Session key diversifier	64 bits	Randomly generated (own) / received from the other device (partner's)	$\mathsf{k}_{\mathrm{AES}}$ generation in BLE	Section 2.3 [Vol 6, Part B, Section 5.1.3.1]
$SRES\_C$	Central's Signed Response	32 bits	32 most significant bits from HMAC(dk,$AU\_RAND\_C|AU\_RAND\_P$)	As a response in the challenge-response scheme in BR/EDR Authentication	Section 2.3 [Vol 2, Part H, Section 7.7.8]
$SRES\_P$	Peripheral's Signed Response	32 bits	33-64th most significant bits from HMAC(dk,$AU\_RAND\_C|AU\_RAND\_P$)	As a response in the challenge-response scheme in BR/EDR Authentication	Section 2.3 [Vol 2, Part H, Section 7.7.8]
TxAdd	Transmission Address Type	1 bit	0 if BD_ADDR is public, 1 if BD_ADDR is random	Indication of the sender's address type in BLE	Section 5.1 [Vol 6, Part B, Section 2.3]
TxAddA	Transmission Address Type of device A	1 bit	Known by a device (if A) or received during radio link establishment (if B)	Indication of BD_ADDR$_A$ address type in BLE	Section 5.1 [Vol 6, Part B, Section 2.3]
TxAddB	Transmission Address Type of device B	1 bit	Known by a device (if B) or received during radio link establishment (if A)	Indication of BD_ADDR$_B$ address type in BLE	Section 5.1 [Vol 6, Part B, Section 2.3]

Variable	Name	Size	Value/Origin	Used in	References
Va	Verification value on device A In [BT523]: also confirmation value	32 bits	32 least significant bits from: BR/EDR: SHA($[g^a]_x$, $[g^b]_x$, Na, Nb) BLE: CMAC($[g^a]_x$, $[g^b]_x$, Na, Nb)	Displayed decimal digits for a user to confirm in NumCom	[Vol 2, Part H, Section 7.2.1] [Vol 2, Part H, Section 2.3.5.6] [Vol 2, Part H, Section 7.7.2] [Vol 3, Part H, Section 2.2.9]
Vb	Verification value on device B In [BT523]: also confirmation value	32 bits	32 least significant bits from: BR/EDR: SHA($[g^a]_x$, $[g^b]_x$, Na, Nb) BLE: CMAC($[g^a]_x$, $[g^b]_x$, Na, Nb)	Displayed decimal digits for a user to confirm in NumCom	[Vol 2, Part H, Section 7.2.1] [Vol 2, Part H, Section 2.3.5.6] [Vol 2, Part H, Section 7.7.2] [Vol 3, Part H, Section 2.2.9]