Cryptographic Analysis of the Bluetooth Secure Connection Protocol Suite

Marc Fischlin and Olga Sanina

Cryptoplexity, Technische Universität Darmstadt, Germany
www.cryptoplexity.de
first.last@tu-darmstadt.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.

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 5.2 [9].

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).[1]

[1] Strictly speaking, there is another mode, the AMP (Alternative MAC/PHY) alias HS (high speed) mode, which is also associated to the classical version. We follow the common terminology to call the classical Bluetooth protocol BR/EDR instead of BR/EDR/AMP.

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 a channel key for communication following the link establishment and to authenticate devices and derive a new channel key in later reconnections. In the latest version 5.2 of the standard [9], 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 methods are nowadays called legacy protocols.

We note that 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 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 should 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.

The first three modes (NumCom, PasskeyEntry, and OOB) are referred to as authenticated, whereas the JustWorks mode is called unauthenticated in the Bluetooth standard [9].

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. [27], for example, the PasskeyEntry method is susceptible to man-in-the-middle attacks. It is based on the different input/output capabilities of devices. In the attack, 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 to authenticate devices reliably.

With the Bluetooth Impersonation AttacksS (BIAS) Antonioli et al. [1] have demonstrated that an adversary can enforce a reconnection for classic Bluetooth to any of two parties sharing a link key, without the adversary actually knowing the key. The attack exploits that legacy authentication of BR/EDR does not enforce mutual authentication of partners and that the request to switch master and slave role is not protected under the shared key. If this is case, then 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 downgrades to legacy security because the request is not authenticated.

Another problem with the PasskeyEntry protocol has been pointed out by Troncoso and Hale [23]. They discuss that the initiator- or responder-generated passkey protocol allows a man-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.

The recent paper of von Tschirschnitz et al. [24] introduced the Method Confusion Attack, which allows the adversary to place itself in the middle between two devices. The adversary establishes two connections with the devices by running the PasskeyEntry mode in one session and the NumCom mode in the other one. Since it can ask the user in the first connection (PasskeyEntry mode) to enter exactly the value used in the second connection (NumCom mode), 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 modes.

Another active attack on the initial connection has been presented by Claverie and Lopes-Esteve [12], called BlueMirror. In this attack, the adversary mounts a man-in-the-middle attack on the passkey subprotocol, reflecting the data in the execution with the initiator, and eventually making the responder believe to communicate with the original initiator. Still, the adversary holds the key in the execution with the responder.

The bad interplay of Bluetooth Classic and Bluetooth Low energy has been exploited in the so-called BLUR attack [3]. If the devices establish a key in the classic or in the low-energy 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 [3], 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 [4,2] where the man-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 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.

Another downgrade attack is the Bluetooth LE Spoofing Attack (BLESA), described in [26]. The attack comes in two versions and has also been shown feasible in practice. One attack version of BLESA is on reactive authentication and lets the adversary make the partner device switch to an encryption-free 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, the latter in the implementations.

We conclude this section by noting that, so far, the OOB mode 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 modes (e.g., associating PasskeyEntry and NumCom in the Method Confusion Attack [24]), or between the classic and low energy cross-modes (like the BLUR attack [3]), or forcing the devices to switch to weak legacy modes (like the BIAS attack [1]). In contrast, most cryptographic analysis focus on a single mode only.

In [19] 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 [21] 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 [19] 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 [23] in the attack section above. Noting the insecurities in the PasskeyEntry sub protocol, 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 [17].

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 [23])—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 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 methods 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 analyses assumes to be "close to the standard". For instance, the security analyses in [19,21,23] assume that the parties use a fresh Diffie-Hellman share in each execution. The Bluetooth v5.2 standard, however, allows the Diffie-Hellman key to be re-used in several executions [9, 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 step of the SSP protocol [9, Vol 2, Part H, Section 7.1]. In particular, in [19] 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 [19,21] 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 [7], Troncoso and Hale [23] correctly use the x-coordinate in some of the protocol steps.

1.5 Privacy

Bluetooth Low Energy supports a privacy mechanism that should help to disguise the device's Bluetooth address `BD_ADDR` 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 address.

In contrast, classic Bluetooth does not support address randomization or any other other privacy mechanism. According to [14], 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 [14]. 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. [22] provide an analysis of the BLE protocol, pointing out correctly that re-using 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 pairing stage only.

Of course, besides inspecting the payload, an attacker may be able to distinguish devices according to physical characteristics. This question recently gained attention in light of contact tracing via Bluetooth. For instance, Ludant et al. [20] showed that dual-mode devices supporting classic Bluetooth (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 services with high accuracy. This implies that the privacy mechanism of BLE effectively becomes void because of the lack of privacy for classic Bluetooth. Countermeasures may be to temporarily turn off either of the two unused protocols or to reduce the transmission power in order to limit the attack radius.

Jouans et al. [18] 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 [10] 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 [25] and BLE [5], it has been pointed out in [11] 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 clear 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.

2 Bluetooth

We start by giving an overview over the Bluetooth protocol along the standard [9]. The Bluetooth protocol comes in several versions with minor differences.

The most common protocols 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 this computation. In the reconnection step, however, the two protocols 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 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 subprotocols 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 1. 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. 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.

Then the devices connect on the link layer and can start exchanging device-specific information, especially the input/output capabilities. Here 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).

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

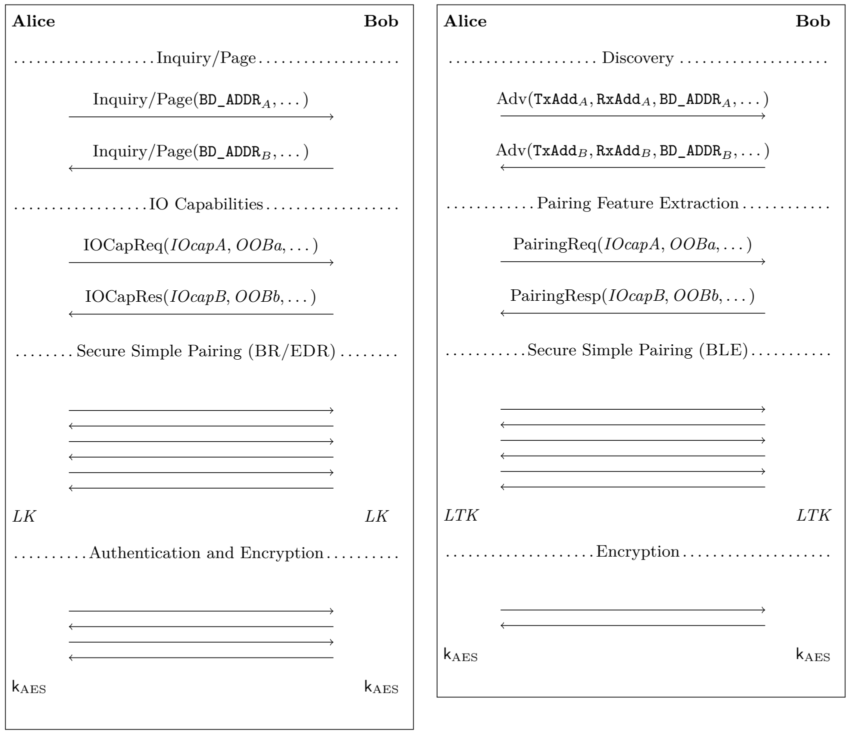

Fig. 1: Bluetooth Protocol Flow (left: BR/EDR, right: BLE)

this is a flag in the pairing features. If either devices 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 transmit 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 to convert the key for future use in the other type of connection (cross-transport key derivation), but we do not consider this conversion 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 if the devices have bonded and created a shared key (i.e. during reconnection), and in this case they 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.

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

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 *IOcap* 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.

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. For the MAC key computation we note that 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.

Figure 2 shows the Numeric Comparison protocol with the abstract operations. The NumCom protocol starts with the devices exchanging the Diffie-Hellman values, 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 the 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

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	$\mathsf{CMAC}(T, 0x00 \mid \mathrm{kID}_{\mathrm{BLE}} \mid N1 \mid N2 \mid A1 \mid A2 \mid 0x0100)$
$\mathsf{MAC}(W, N1, N2, R, I, A1, A2)$	$\mathsf{HMAC}(W, N1 \mid N2 \mid R \mid I \mid A1 \mid A2)/2^{128}$	$\mathsf{CMAC}(W, N1 \mid N2 \mid I \mid A1' \mid A2')$
$\mathsf{KDF}(W, N1, N2, A1, A2)$	$\mathsf{HMAC}(W, N1 \mid N2 \mid \mathrm{kID}_{\mathrm{BR/EDR}} \mid A1 \mid A2)/2^{128}$	$\mathsf{CMAC}(T, 0x01 \mid \mathrm{kID}_{\mathrm{BLE}} \mid N1 \mid N2 \mid A1 \mid A2 \mid 0x0100)$

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 the full version. 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.

2.3 Deriving the Encryption Key

The encryption key is derived differently in classic Bluetooth and in the Low Energy version. In the classic 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) for authentication, 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 also 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 IVm, IVs, 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 3, and for BLE in Figure 4. We use the common notation of *master* and *slave* since the devices may change roles for reconnections. We note that in BLE the key derivation step and

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

Fig. 2: Bluetooth Secure Simple Pairing in mode Numeric Comparison. The session identifier, here and in all other association models, is given by $\mathsf{sid} = (g^a, g^b, A, B, Na, Nb)$.

the data (*SKDm*,*IVm* resp. *SKDs*,*IVs*) are transmitted as part of an encryption request and response message. In BR/EDR the sequence must be preceded by an encryption_mode request and response. Noteworthy, 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 attacks like the KNOB attack [4,2].

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

Table 3: Secure Authentication and Computation of Encryption Key in BR/EDR Secure Connections. HMAC is HMAC with SHA256; $\mathrm{kID}_{\mathrm{Dev}} = 0x6274646B$ is a 4-octet representing the ASCII string 'btdk' (Bluetooth Device Key); $\mathrm{kID}_{\mathrm{AES}} = 0x6274616B$ is a 4-octet representing the ASCII string 'btak' (Bluetooth AES Key); $SRESm, SRESs$ 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	$\mathrm{dk} \leftarrow \mathsf{HMAC}(LK, \mathrm{kID}_{\mathrm{Dev}}\vert\mathtt{BD\_ADDR}_A\vert\mathtt{BD\_ADDR}_B)/2^{128}$
Confirmation	$SRESm\vert SRESs\vert\mathrm{ACO} \leftarrow \mathsf{HMAC}(\mathrm{dk}, AU\_RANDm\vert AU\_RANDs)/2^{128}$
AES Key	$\mathsf{k}_{\mathrm{AES}} \leftarrow \mathsf{HMAC}(LK, \mathrm{kID}_{\mathrm{AES}}\vert\mathtt{BD\_ADDR}_A\vert\mathtt{BD\_ADDR}_B\vert\mathrm{ACO})/2^{128}$

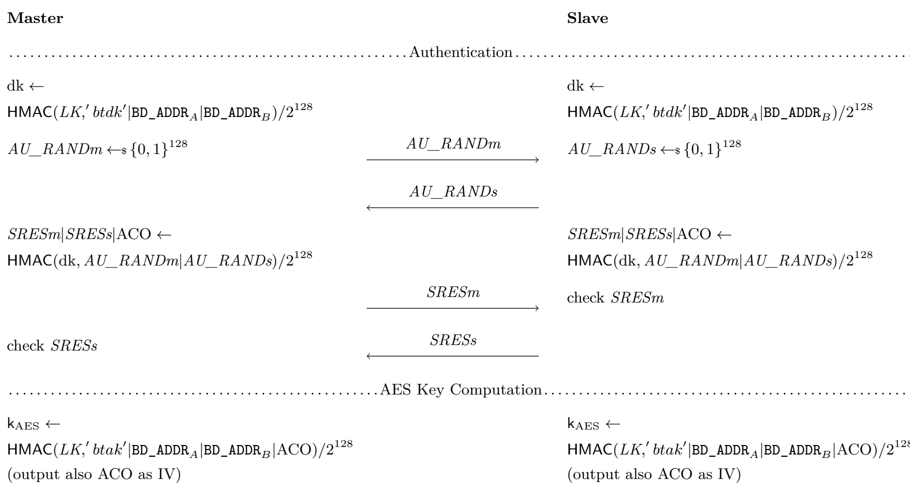

Fig. 3: Bluetooth BR/EDR Secure Authentication and Encryption Key Derivation. The session identifier for this subprotocol is given by $\mathsf{sid} = (AU\_RANDm, AU\_RANDs)$.

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 a 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: isTested for session keys which have been tested,

Master		**Slave**
$SKDm \leftarrow_{\$} \{0,1\}^{64}, IVm \leftarrow_{\$} \{0,1\}^{32}$		$SKDs \leftarrow_{\$} \{0,1\}^{64}, IVs \leftarrow_{\$} \{0,1\}^{32}$
	$\xrightarrow{SKDm, IVm}$	
	$\xleftarrow{SKDs, IVs}$	
$\mathsf{k}_{\text{AES}} \leftarrow \text{AES}(LTK, SKDm\|SKDs)$		$\mathsf{k}_{\text{AES}} \leftarrow \text{AES}(LTK, SKDm\|SKDs)$
(output also $IVm\|IVs$ as IV)		(output also $IVm\|IVs$ as IV)

Fig. 4: Bluetooth BLE Encryption Key Derivation. The session identifier is given as $\mathsf{sid} = (SKDm, SKDs)$.

$\mathsf{isRevealed}$ for session keys which have been revealed, and isTOFU for session keys which 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 [6]. We assume that parties have some identity. For Bluetooth this will be the 48-bits Bluetooth device address `BD_ADDR` of the device, which can be either public or random. According to the Bluetooth protocol description we sometimes denote the identities of connecting devices as A and B. Parties know their identity and also know 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 re-use secret information like the Diffie-Hellman shares in multiple executions, we need to account for such attack vectors.

For the re-usable 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 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 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 which should be used, and further data like $\mathsf{passkey} \in \{0, 1, \ldots, 9\}^* \cup \{\perp\}$ in the passkey entry mode or information transmitted out of band.
- LinkKey describes the connection or link 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 running, or has accepted or rejected.
- The Boolean variable isTested determines if the session key has been tested before. Initialized to false.
- The Boolean variable 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 coincides with the connection key.
- $\mathsf{sid} \in \{0, 1\}^* \cup \{\perp\}$ 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 1 (Partnered Sessions). *We say that two sessions* $\mathsf{lbl}, \mathsf{lbl}'$ *are partnered if* $\mathsf{lbl} \neq \mathsf{lbl}'$ *and* $\mathsf{lbl.sid} = \mathsf{lbl}'.\mathsf{sid} \neq \perp$.

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., $\mathsf{lbl.id} \leftarrow i$, the mode is set to $\mathsf{lbl.mode} \leftarrow \mathsf{init}$, and the optional parameter [aux], if present, is stored in $\mathsf{lbl.aux}$ (and otherwise this entry is set to $\perp$). We set $\mathsf{lbl.state} \leftarrow \mathsf{running}$, $\mathsf{lbl.pkctr} \leftarrow \mathsf{pkctr}_i$, as well as $\mathsf{lbl.isTested}, \mathsf{lbl.isRevealed}, \mathsf{lbl.isTOFU} \leftarrow \mathsf{false}$, 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 $\mathsf{lbl.LinkKey} \neq \perp$. 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 $\mathsf{lbl}'.\mathsf{isTOFU} \leftarrow \mathsf{lbl.isTOFU}$, and copies the previous connection key, $\mathsf{lbl}'.\mathsf{LinkKey} \leftarrow \mathsf{lbl.LinkKey}$. 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 LinkKey-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 2 (Match-Security). *We say that a key exchange protocol Π provides Match-security if for any PPT adversary $\mathcal{A}$ and identity set $\mathcal{I}$ we have*

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

is negligible, where

$\boldsymbol{Exp}^{Match}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda)$

$b \leftarrow_{\$} \{0,1\}$
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 $\mathsf{lbl.LinkKey} = \mathsf{lbl}'.\mathsf{LinkKey}$ ***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 which 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 $\mathsf{isTOFU} = \mathsf{false}$).

Definition 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

$\boldsymbol{Exp}^{Secrecy}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda)$

$b \leftarrow_{\$} \{0,1\}$
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) provides 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. 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 re-used 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 [8] 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.

Definition 4 (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 $\mathsf{PRF}(X^w, s)$.*

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

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

is negligible, where

$\mathbf{Exp}^{PRF\text{-}ODH}_{\mathcal{A},\mathsf{PRF},\mathbb{G}}$

$u, v \leftarrow_\$ \mathbb{Z}_q, b \leftarrow_\$ \{0,1\}$
$U \leftarrow g^u, V \leftarrow g^v$
$(x^*, st) \leftarrow_\$ \mathcal{A}^{\mathsf{ODH}_u(\cdot,\cdot),\mathsf{ODH}_v(\cdot,\cdot)}(U, V)$
$y_0 \leftarrow \mathsf{PRF}(g^{uv}, x^*), y_1 \leftarrow \{0,1\}^{|y_0|}$
$a \leftarrow_\$ \mathcal{A}^{\mathsf{ODH}_u(\cdot,\cdot),\mathsf{ODH}_v(\cdot,\cdot)}(st, V, y_b)$
return $a = b$

where we assume that $\mathcal{A}$ never makes a query $(A, x) = (V, x^*)$ to oracle ODH_u resp. $(B, x) = (U, x^*)$ 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, $\mathsf{PRF}(W, x) = \mathsf{CMAC}(\mathsf{CMAC}(\mathrm{Salt}, W), x)$. 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 [7] 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 which oracle is used made at a random.

4.2 Match Security

We first argue Match-security of the Bluetooth protocol. Recall that we set the session identifiers to consist of $\mathsf{sid} = (g^a, g^b, A, B, Na, Nb)$ for the initial connection, and $\mathsf{sid} = (AU\_RANDm, AU\_RANDs, A, B)$ for BR/EDR reconnections resp. $\mathsf{sid} = (SKDm, SKDs)$ for BLE. 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 LinkKey of the session.

Proposition 1 (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 $\mathsf{KDF}(g^{ab}, Na, Nb, A, B)$ such that the output of the (deterministic) key derivation matches for equal session identifiers. 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 $(AU\_RANDm, AU\_RANDs, A, B)$ resp. $(SKDm, SKDs)$ 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_RANDm, AU_RANDs for reconnections in BR/EDR resp. 64-bit values $SKDm, SKDs$ in BLE. it follows that each pairs yields a nonce collision with probability at most $2^{-|\mathrm{nonce}|} = 2^{-128}$ in BR/EDR resp. $\leq 2^{-64}$ in BLE. The overall threefold collision probability for session identifiers is thus at most $q_s^2 \cdot 2^{-|\mathrm{nonce}|}$ as stated. □

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 2 (Key Secrecy). *The Bluetooth protocol Π provides trust-on-first-use Secrecy. That is, for any adversary $\mathcal{A}$ initiating at most q_s sessions there exists adversaries $\mathcal{B}$ and $\mathcal{C}$ (with roughly the same run time as $\mathcal{A}$, and $\mathcal{C}$ making at most q_s oracle queries) such that*

$$\boldsymbol{Adv}^{Secrecy}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda) \leq q_s^3 \cdot \boldsymbol{Adv}^{PRF\text{-}ODH}_{\mathcal{B},\mathsf{PRF},\mathbb{G}}(\lambda) + q_s \cdot \boldsymbol{Adv}^{PRF}_{\mathcal{C},\mathsf{PRF}'}(\lambda) + q_s^2 \cdot 2^{-|nonce|}.$$

where $|nonce| = 128$*, and* 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*.*

We note that the reduction factor q_s^3 is indeed large but follows other analyses. A factor q_s comes from the multiple test queries which our model allows, and the quadratic term q_s^2 from the need to guess the correct insertion points of the Diffie-Hellman keys. For instance, Troncoso and Hale [23] 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 [16] or to use and program a random oracle [13,15]. 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_s^2 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, $\Pr[\mathsf{Game}_0] = \mathbf{Exp}^{\mathit{Secrecy}}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda) - \frac{1}{2}$.

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 $q_s^2 \cdot 2^{-|\text{nonce}|}$. 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 $\mathsf{LinkKey}$ in each session lbl in mode $\mathsf{lbl.mode} = \mathsf{init}$ upon acceptance as follows: If there is a partnered session lbl' which has accepted before—there can be at most one by the previous game hop—set $\mathsf{lbl.LinkKey} \leftarrow \mathsf{lbl}'.\mathsf{LinkKey}$. Else, replace $\mathsf{lbl.LinkKey}$ by a random string of the same length.

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 $\mathsf{LinkKey}$ by construction.

We argue that $\Pr[\mathsf{Game}_1] \leq \Pr[\mathsf{Game}_2] + q_s^3 \cdot \mathbf{Adv}^{\text{PRF-ODH}}_{\mathcal{B},\mathsf{PRF},\mathbb{G}}(\lambda)$. The argument is via an (interactive) hybrid argument against the PRF-ODH assumption. Details are omitted here for space reasons; they appear in the full version.

Game 3. In Game_3 we can now replace all session keys in sessions $\mathsf{lbl.mode} = \mathsf{reconnect}$ and $\mathsf{lbl.isTOFU} = \mathsf{true}$ by random values, ignoring any consistency requirement.

Note that such sessions are exactly those where we have replaced the connection key LinkKey by a fresh random value. Also observe that the security game ensures that the key of the partner session of a revealed session key or any tested session key cannot be obtained again, such that we do not need to take care of consistency here. It follows now via a straightforward reduction to the pseudorandomness of HMAC resp. AES, with a hybrid argument over all at most q_s connection keys, that this is indistinguishable from the adversary's point of view.

In game Game_3 the adversary gets to see a random and independent session key in either of the two cases of the challenge bit b. Hence, the probability of predicting b correctly is exactly $\frac{1}{2}$. The claim now follows from collecting all probabilities. □

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.

Private Addresses. To support the privacy mechanism, the standard specifies four types of Bluetooth addresses BD_ADDR 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 [9, 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 *prand* that are set randomly—effectively only 22 random bits since the most significant bits correspond to '10'— 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.

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 $IRK \leftarrow \mathsf{AES}(IR, 0x00000000|0x01|0x00)$. Noteworthy, 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:

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

where the 24-bit value *prand* consists of the 22 random bits and '10'. 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. First the slave sends its *IRK*, address, and *CSRK*. Then the exchange is followed by the master 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 master is called the initiator, and the link layer slave 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 [4,2] where the man-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 man-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 master or the slave 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 [9, 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 (Outsider Privacy). *The key exchange protocol Π provides outsider privacy if for any PPT adversary $\mathcal{A}$*

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

is negligible, where

$$\begin{array}{l}\underline{\boldsymbol{Exp}^{Privacy}_{\mathcal{A},\Pi,\mathcal{I}}(\lambda)}\\ b \leftarrow_{\$} \{0,1\}\\ \textbf{\textit{forall}}\ i \in \mathcal{I}\ \textbf{do}\\ \quad IRK \leftarrow_{\$} \{0,1\}^{128} \setminus \{0\}\\ a \leftarrow_{\$} \mathcal{A}^{\mathsf{Test}}(\mathcal{I})\\ \textbf{return}\ \ 1\ \textbf{if}\ a = b\end{array}$$

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 3 (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*

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

where $|prand| = 24$.

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 $q_s^2 \cdot 2^{-|prand|+2}$. 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 3). 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 $\mathsf{AES}(IRK, \cdot)$ 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 *IRK*s by looking at the identities of devices. This step occurs a loss of $q_s \cdot \mathbf{Adv}^{\mathrm{PRF}}_{\mathcal{B},\mathsf{AES}}(\lambda)$, 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 capablities 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 $SKDm, IVm$ and $SKDs, IVs$. 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 is 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 man-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

1. Antonioli, D., Tippenhauer, N.O., Rasmussen, K.: BIAS: Bluetooth impersonation AttackS. In: 2020 IEEE Symposium on Security and Privacy. pp. 549–562. IEEE Computer Society Press, San Francisco, CA, USA (May 18–21, 2020)
2. Antonioli, D., Tippenhauer, N.O., Rasmussen, K.: Key negotiation downgrade attacks on bluetooth and bluetooth low energy. ACM Trans. Priv. Secur. 23(3), 14:1–14:28 (2020), `https://doi.org/10.1145/3394497`
3. Antonioli, D., Tippenhauer, N.O., Rasmussen, K., Payer, M.: BLURtooth: Exploiting Cross-Transport Key Derivation in Bluetooth Classic and Bluetooth Low Energy (2020)
4. Antonioli, D., Tippenhauer, N.O., Rasmussen, K.B.: The KNOB is broken: Exploiting low entropy in the encryption key negotiation of bluetooth BR/EDR. In: Heninger, N., Traynor, P. (eds.) USENIX Security 2019: 28th USENIX Security Symposium. pp. 1047–1061. USENIX Association, Santa Clara, CA, USA (Aug 14–16, 2019)
5. Becker, J.K., Li, D., Starobinski, D.: Tracking anonymized bluetooth devices. Proceedings on Privacy Enhancing Technologies 2019(3), 50–65 (Jul 2019)
6. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) Advances in Cryptology – CRYPTO'93. Lecture Notes in Computer Science, vol. 773, pp. 232–249. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 22–26, 1994)
7. Biham, E., Neumann, L.: Breaking the bluetooth pairing - the fixed coordinate invalid curve attack. In: Paterson, K.G., Stebila, D. (eds.) SAC 2019: 26th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 11959, pp. 250–273. Springer, Heidelberg, Germany, Waterloo, ON, Canada (Aug 12–16, 2019)

8. Brendel, J., Fischlin, M., Günther, F., Janson, C.: PRF-ODH: Relations, instantiations, and impossibility results. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017, Part III. Lecture Notes in Computer Science, vol. 10403, pp. 651–681. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2017)
9. Bluetooth Core Specification (December 2019), ver. 5.2.
10. Celosia, G., Cunche, M.: Fingerprinting bluetooth-low-energy devices based on the generic attribute profile. In: Liu, P., Zhang, Y. (eds.) 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. pp. 24–31. ACM (2019), https://doi.org/10.1145/3338507.3358617
11. Celosia, G., Cunche, M.: Saving private addresses: an analysis of privacy issues in the bluetooth-low-energy advertising mechanism. In: Poor, H.V., Han, Z., Pompili, D., Sun, Z., Pan, M. (eds.) 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. pp. 444–453. ACM (2019), https://doi.org/10.1145/3360774.3360777
12. Claverie, T., Lopes-Esteves, J.: Bluemirror: Reflections on bluetooth pairing and provisioning protocols. In: IEEE Security and Privacy Workshops, SP Workshops 2021, San Francisco, CA, USA, May 27, 2021. pp. 339–351. IEEE (2021), https://doi.org/10.1109/SPW53761.2021.00054
13. Cohn-Gordon, K., Cremers, C., Gjøsteen, K., Jacobsen, H., Jager, T.: Highly efficient key exchange protocols with optimal tightness. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology – CRYPTO 2019, Part III. Lecture Notes in Computer Science, vol. 11694, pp. 767–797. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019)
14. Cominelli, M., Gringoli, F., Patras, P., Lind, M., Noubir, G.: 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. pp. 534–548. IEEE (2020), https://doi.org/10.1109/SP40000.2020.00091
15. Diemert, D., Jager, T.: On the tight security of TLS 1.3: Theoretically-sound cryptographic parameters for real-world deployments. Cryptology ePrint Archive, Report 2020/726 (2020), https://eprint.iacr.org/2020/726
16. Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 95–125. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018)
17. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 273–293. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012)
18. Jouans, L., Viana, A.C., Achir, N., Fladenmuller, A.: 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. pp. 1–6. IEEE (2021), https://doi.org/10.1109/CCNC49032.2021.9369628
19. Lindell, A.Y.: Comparison-based key exchange and the security of the numeric comparison mode in Bluetooth v2.1. In: Fischlin, M. (ed.) Topics in Cryptology –

CT-RSA 2009. Lecture Notes in Computer Science, vol. 5473, pp. 66–83. Springer, Heidelberg, Germany, San Francisco, CA, USA (Apr 20–24, 2009)

20. Ludant, N., Vo-Huu, T.D., Narain, S., Noubir, G.: 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)
21. Sun, D.Z., Sun, L.: 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), https://www.mdpi.com/1424-8220/19/5/1158
22. Sun, D., Sun, L., Yang, Y.: On secure simple pairing in bluetooth standard v5.0-part II: privacy analysis and enhancement for low energy. Sensors 19(15), 3259 (2019), https://doi.org/10.3390/s19153259
23. Troncoso, M., Hale, B.: The bluetooth cyborg: Analysis of the full human-machine passkey entry ake protocol. Cryptology ePrint Archive, Report 2021/083 (2021), https://eprint.iacr.org/2021/083
24. von Tschirschnitz, M., Peuckert, L., Franzen, F., Grossklags, J.: Method confusion attack on bluetooth pairing. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 213–228. IEEE Computer Society, Los Alamitos, CA, USA (may 2021), https://doi.ieeecomputersociety.org/10.1109/SP40001.2021.00013
25. Vanhoef, M., Matte, C., Cunche, M., Cardoso, L.S., Piessens, F.: Why MAC address randomization is not enough: An analysis of wi-fi network discovery mechanisms. In: Chen, X., Wang, X., Huang, X. (eds.) Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016. pp. 413–424. ACM (2016), https://doi.org/10.1145/2897845.2897883
26. Wu, J., Nan, Y., Kumar, V., Tian, D.J., Bianchi, A., Payer, M., Xu, D.: BLESA: Spoofing attacks against reconnections in bluetooth low energy. In: 14th USENIX Workshop on Offensive Technologies (WOOT 20). USENIX Association (Aug 2020), https://www.usenix.org/conference/woot20/presentation/wu
27. Zhang, Y., Weng, J., Dey, R., Jin, Y., Lin, Z., Fu, X.: Breaking secure pairing of bluetooth low energy using downgrade attacks. In: Capkun, S., Roesner, F. (eds.) USENIX Security 2020: 29th USENIX Security Symposium. pp. 37–54. USENIX Association (Aug 12–14, 2020)