Noname manuscript No.
(will be inserted by the editor)

Analyzing the Secure Simple Pairing in Bluetooth v4.0

Raphael C.-W. Phan · Patrick Mingard

Received: 27 January 2010, Revised: 3 December 2010

Abstract This paper analyzes the security of Bluetooth v4.0's Secure Simple Pairing (SSP) protocol, for both the Bluetooth Basic Rate / Enhanced Data Rate (BR/EDR) and Bluetooth Low Energy (LE) operational modes. Bluetooth v4.0 is the latest version of a wireless communication standard for low-speed and low-range data transfer among devices in a human's PAN. It allows increased network mobility among devices such as headsets, PDAs, wireless keyboards and mice. A pairing process is initiated when two devices desire to communicate, and this pairing needs to correctly authenticate devices so that a secret link key is established for secure communication. What is interesting is that device authentication relies on humans to communicate verification information between devices via a human-aided out-of-band channel. Bluetooth v4.0's SSP protocol is designed to offer security against passive eavesdropping and man-in-the-middle (MitM) attacks. We conduct the first known detailed analysis of SSP for all its MitM-secure models. We highlight some issues related to exchange of public keys and use of the passkey in its models and discuss how to treat them properly.

Keywords Bluetooth v4.0 · Low Energy (LE) · Secure Simple Pairing · association models

1 Introduction

Wireless personal communication networks based on mobile lightweight devices like smart cards, RFIDs, PDAs, sensors and mobile agents require innovative ways to au-

Part of this work done while the authors were with École Polytechnique Fédérale de Lausanne (EPFL), Switzerland.

R. Phan
Electronic & Electrical Engineering, Loughborough University, UK
Tel.: +44-1509-227086
Fax: +44-1509-227014
E-mail: r.phan@lboro.ac.uk

P. Mingard
La Mobilière, Lausanne, Switzerland
E-mail: patrick.mingard@a3.epfl.ch

thenticate and secure the wireless communications among devices. One of the methods increasingly being deployed for this is to leverage on human users to aid in authenticating their devices with other peer devices.

This paper concentrates on Bluetooth v4.0, a wireless communication standard for low-speed and low-range data transfer for personal area networks (PANs). It uses the unlicensed radio frequency of 2.4 GHz. The main goal of Bluetooth is to replace a wired connection with a wireless one for the ease of the users. Well known applications are headsets for mobile cell phones, gaming consoles, phone-to-phone image and music exchange or wireless keyboards and mice. A Bluetooth device can be either master or slave and the set of interconnected devices is called a "piconet". The master device can be connected up to seven slaves while slaves can be connected to only one master.

When two devices desire to communicate, they need to first authenticate each other and then establish a shared secret link key LK that is used to secure subsequent wireless communications between them; and the way it is done is via humans as middle persons via physical so-called out-of-band channels; similar human-aided authentication protocols exist e.g. those [28] based on voice-over-IP. The authentication process is called *pairing* and the latest pairing protocol for Bluetooth v4.0 is so-called the Secure Simple Pairing (SSP) protocol [3]. Indeed, including the human-based processing element into mobile networks such as Bluetooth adds an extra level of trust that can be used for authenticating Bluetooth devices: human users who own the devices desire to pair their devices with peers, so their behaviour can be trusted more compared to depending solely on the devices to authenticate without human aid.

We analyze the security of SSP in detail, in the context of all its association models, for both the Bluetooth v4.0's Basic Rate/Enhanced Data Rate (BR/EDR) and Low Energy (LE) operation operational modes. The Bluetooth LE is new to v4.0 and its corresponding security association modes differ slightly from those of Bluetooth BR/EDR.

The SSP is in fact similar to authentication [23,29] and key establishment (AKE) [24] protocols, and therefore we consider the following basic security properties required of such protocols [20].

- Known key security (KKS) [20]: Compromising a session key does not leak out other session keys.
- Key control (KC) [20,22]: No device should be able to influence a link key to some biased value.
- Perfect forward secrecy (PFS) [20]: If long-term secrets or private keys of any device are compromised, the secrecy of previously established session keys should not be affected. This attempts to still offer some security guarantee in spite of the fact that the long-term secret has been leaked.
- Key-compromise impersonation (KCI) resilience [14,4]: The compromise of any device A's long-term key or secret should not enable the attacker to impersonate any other devices to A.
- Unknown key-share attack (UKS) resilience [6,20]: UKS is an attack where a device A believes that it shares a key with another device B upon completion of a protocol run (this is in fact the case), but B falsely believes that the key is instead shared with a device $E \neq A$.
- Man-in-the-Middle (MitM) attack resilience: For any protocol where devices desire to authenticate each other, it should not be possible for an attacker to place himself in the middle of the two devices and cause them to have false beliefs about how

the protocol actually executed. This is one of the main properties that the SSP was designed to offer, and as we will demonstrate .

- Offline dictionary attack resilience: Since one of SSP's association modes uses a passkey, bearing similarities to a password in the context of password-based AKE protocols, we list security against offline dictionary attacks here for completeness. Originally, a dictionary attack is a password guessing technique in which the attacker attempts to determine a user's password by successively trying words from a dictionary (a compiled list of likely passwords) in the hope that one of these password guesses will be the user's actual password. A dictionary attack that can be performed offline i.e. without having to actively interact with any device within a live protocol session, is devastating. All password-based protocols should be resilient to this type of attack.

1.1 Related Work

Earlier parts of our attacks [21] on SSP concentrated on its PE model. The Bluetooth version we analyze in this paper is the latest specification, v4.0. New to this version is the Bluetooth LE operational mode, with slightly different security association models. To the best of our knowledge, this paper is the first known security analysis of v4.0 and Bluetooth LE in detail for all their models that provide security against MitM attacks.

Kuo et al. [16] briefly overviews potential general security risks of Bluetooth considering that it supports multiple setup mechanisms. A remark about a security issue for SSP in the Passkey Entry model in terms of passkey leakage was also mentioned en passant therein, as was later independently observed by Lindell [17,18]. Their observations relate to potential password leakage, yet the passkey in the Bluetooth context is different from a conventional password in that a passkey is mainly used for authentication rather than secrecy; and its existence is to add an extra security factor to the pairing authentication between Bluetooth parties. Thus, even if the passkey is guessable, SSP in the PE model remains secure unless all of its stages are vulnerable to attack. Suomalainen et al. [26] gives a comparative overview of different pairing models in personal networks including Bluetooth. They identify as a potential attack scenario where the security of a more IO-capable device is compromised by having it interact with another device of restricted IO-capability e.g. one without display capability. Chang and Shmatikov [5] applied a formal methods tool to analyze the authentication aspect of SSP in the numeric comparison model, and showed that if the same device is used concurrently in different sessions then authentication fails. This is because even if the user correctly checks that the numbers displayed on both devices are equal, they may not necessarily be involved in the same intended session. Haataja et al. [10,11,8,9] exploited the fact that prior to SSP the devices exchange their respective input/output capabilities without any authentication, and so describe that one could modify these exchange messages to force devices to use the Just Works (JW) association model whose SSP is not designed to resist MitM attacks, thus leading to an MitM attack on the devices.

2 Bluetooth's Secure Simple Pairing (SSP) protocol

The latest Bluetooth Core specification is version 4.0, officially published 30 June 2010. Bluetooth v4.0 includes the Secure Simple Pairing (SSP) protocol [3] that specifies how two Bluetooth devices establish a shared secret link key LK for subsequent secure communication. It also specifies the SSP protocol variants for the new Bluetooth Low Energy (LE) operational mode.

Bluetooth devices can have a range of input and output (IO) resources, from screen and keyboard in a mobile phone to near nothing in a cordless headset. Thus, the way that the SSP is expected to interact with the human user especially for operations involving a short authenticated string (SAS), would differ depending on IO capability of the device. Hence, there are four variants of the SSP protocol, called *association models*, each designed for a different type of device with a particular IO resource:

1. Numeric Comparison (NC)
2. Just Works (JW)
3. Passkey Entry (PE)
4. Out of Band (OoB)

Security Goals. The SSP is intended to meet two requirements: improved security and simplification for the user. Improved security is in terms of resistance against *passive eavesdropping* (related to offline dictionary attacks but not identical) and also against *man-in-the-middle (MitM) attacks*, except for the JW model. We only consider in this paper the case of models designed to be security against MitM attacks, i.e. all SSP models except the JW model. These MitM-secure models are more interesting, since it is one of SSP's explicit design objectives to offer MitM security (versus older Bluetooth pairings), and also since the JW model is not recommended [25] for Bluetooth devices that support SSP.

The basic idea in MitM-resisting models of SSP is to employ a short authenticated string (SAS) [28] that is transmitted even in the clear but via an out-of-band authenticated channel, e.g. voice, visual or some physical channel other than the wireless channel used by Bluetooth. To be able to offer interesting MitM security, it is assumed that the adversary does not have access to the out-of-band channel used to communicate the SAS. For instance in the NC model, if an adversary could see the SAS displayed on a device, then it is trivial to pair the adversary's device with the legitimate device; similarly in the PE model, it will be easy for the adversarial device to pair with legitimate devices if an adversary could access to the SAS that is input by the user into a legitimate device. Thus, throughout this paper, our discussion does not consider these trivial cases. More precisely, the SAS in the NC model is a 6-digit number V_a (resp. V_b) that is a function of some partial transcript of the current protocol session, and this is displayed at each device for a visual inspection of the user: the visual inspection is the out-of-band authenticated channel in this case. The SAS in the PE model is a user-supplied 6-digit number R_a (resp. R_b), either input by the user at both devices or input at one device and displayed at the other for visual inspection. In the former case, the out-of-band authenticated channel is the fact that the user physically types in the same SAS. In the latter case, the out-of-band authenticated channel is again a visual inspection by the user. Finally, the SAS in the OoB model are random 6-digit numbers R_a and R_b, which are communicated between the devices in some out-of-band authenticated channel. In all models, the SAS is subsequently used in computations of the current protocol session.

Association Models. SSP association models have ordered priority levels depending on IO capability of the Bluetooth device. The OoB model is used as long as either device has OoB capability and has received OoB authentication data. Otherwise, the NC model is used if both devices have display and input capabilities. If at least one device does not have input or display capability, then the PE model is used either as option (i) or option (ii), see Section II.C for more details. See Table 1, where we are only interested in settings where authentication, i.e. security against MitM attacks, is provided. For compatibility with the Bluetooth v4.0 specification, we will use similar notations in this paper, as in Table 2.

Table 1 Mapping IO Capability to Association Model

	A: Display Only	A: DisplayYesNo	A: KeyboardOnly
B: DisplayOnly	-	-	PE(ii)
B: DisplayYesNo	-	NC	PE(ii)
B: KeyboardOnly	PE(ii)	PE(ii)	PE(i)

Table 2 Terminology

SK_x	Private key of device X.
PK_x	Public key of device X, and equals $SK_x \cdot G$ for some public elliptic-curve point G.
IO_x	Description of the input/output capabilities of device X.
X	The Bluetooth address of device X (BD_ADDR_x).
$f_1()$	Function used to generate the 128-bit commitments C_a and C_b in stage (2).
$f_2()$	Function used to generate the link key LK in stage (4).
$f_3()$	Function used to generate the check values E_a and E_b in stage (3).
C_{xi}	128-bit commitment value from device X corresponding to iteration i.
E_x	128-bit check value from device X.
N_{xi}	128-bit nonce (random value used only once) generated by device X corresponding to iteration i.
R_x	6-digit (20-bit) passkey input by user to device X.
R_{xi}	ith bit of R_x for $i = 1, \ldots, 20$.
K	Long-term shared key established via elliptic curve based Diffie-Hellman in stage (1).
LK	Link key between device A and B computed in stage (4).
K_{enc}	Encryption key between device A and B computed in stage (5) as a function of K and LK.
$btlk$	A constant string.

2.1 Numeric Comparison (NC)

This model is designed for communication between devices capable of displaying a 6-digit number and have a keyboard with at least the capability for entering "yes" or "no"; for instance, a mobile phone connected to a computer. See Fig. 1 which illustrates the five stages of the SSP protocol in the NC model. The user is shown a 6-digit number (the SAS) on both devices' screen. The user answers "yes" if the two numbers are the same, "no" otherwise.

Initiating device A		Non-initiating device B	
	$\xrightarrow{PK_a}$	$K = SK_b \cdot PK_a \cdot G$	(1)
$K = SK_a \cdot PK_b \cdot G$	$\xleftarrow{PK_b}$		
Select random N_a.		Select random N_b.	(2)
Set $R_a = R_b = 0$.		Set $R_a = R_b = 0$.	
		Compute $C_b = f_1(PK_b, PK_a, N_b, R_b)$.	
	$\xleftarrow{C_b}$		
	$\xrightarrow{N_a}$		
	$\xleftarrow{N_b}$		
Check $C_b = f_1(PK_b, PK_a, N_b, R_a)$.			
If check fails, abort.			
Compute $V_a = g(PK_a, PK_b, N_a, N_b)$.		Compute $V_b = g(PK_a, PK_b, N_a, N_b)$.	
At each device (A resp. B), V_a (resp. V_b) is displayed as a 6-digit number. If "ok" is pressed by user confirming that $V_a = V_b$, then proceed.			
Compute $E_a = f_3(K, N_a, N_b, R_a, IO_a, A, B)$.		Compute $E_b = f_3(K, N_b, N_a, R_b, IO_b, B, A)$.	(3)
	$\xrightarrow{E_a}$		
		Check $E_a = f_3(K, N_a, N_b, R_a, IO_a, A, B)$.	
		If check fails, abort.	
	$\xleftarrow{E_b}$		
Check $E_b = f_3(K, N_b, N_a, R_b, IO_b, B, A)$.			
If check fails, abort.			
At both devices: $LK = f_2(K, N_a, N_b, \text{“}btlk\text{”}, A, B)$			(4)
At both devices: the encryption key K_{enc} is derived based on K and LK. Further messages are encrypted with K_{enc}.			(5)

Fig. 1 Secure Simple Pairing Protocol in NC model

2.2 Passkey Entry (PE)

This model is designed for scenarios where one device has input but no display capability while the other has at least display capability, e.g. a cordless keyboard connected to a computer. So, it is not possible to display a computed SAS on at least one device, instead the user is asked to input the SAS (the *passkey*). There are two options on how the passkey is used in this model, see Figs. 2 and Fig. 3.

- Option (i): user is asked to enter the same passkey of his choice into both devices, or
- Option (ii): user is shown a 6-digit number and asked to enter into the device without display capability.

Indeed, it is safe to say that most Bluetooth applications are expected to employ usage option (ii), since intuitively it is hard to think of an application where the two pairing devices have keypads but no display. In fact, it appears that it is even more common to have pairing between devices that do not have any display nor keypads, i.e. the JW model, e.g. between a headset and a Bluetooth-enabled car [13].

2.3 Out of Band (OoB)

The OoB mechanism is used when in addition to the wireless medium for which a Bluetooth connection is desired, there exists an external medium (so-called out of band) between the two Bluetooth devices, which can be used to exchange the SAS between

Initiating device A		Non-initiating device B	
	$\xrightarrow{PK_a}$	$K = SK_b \cdot PK_a \cdot G$	(1)
$K = SK_a \cdot PK_b \cdot G$	$\xleftarrow{PK_b}$		
At each device (A, B), the user enters the same 6-digit passkey $R_a = R_b$ via the device input. Let R_{ai} be the ith bit of R_a for $i = 1, \ldots, 20$. The rest of this stage is performed 20 times for $i = 1, \ldots, 20$.			(2)
Select random N_{ai}. Compute $C_{ai} = f_1(PK_a, PK_b, N_{ai}, R_{ai})$.		Select random N_{bi}. Compute $C_{bi} = f_1(PK_b, PK_a, N_{bi}, R_{bi})$.	
	$\xrightarrow{C_{ai}}$		
	$\xleftarrow{C_{bi}}$		
	$\xrightarrow{N_{ai}}$		
		Check $C_{ai} = f_1(PK_a, PK_b, N_{ai}, R_{bi})$. If check fails, abort.	
	$\xleftarrow{N_{bi}}$		
Check $C_{bi} = f_1(PK_b, PK_a, N_{bi}, R_{ai})$. If check fails, abort.			
Set $N_a = N_{a20}$. Compute $E_a = f_3(K, N_a, N_b, R_a, IO_a, A, B)$.		Set $N_b = N_{b20}$. Compute $E_b = f_3(K, N_b, N_a, R_b, IO_b, B, A)$.	(3)
	$\xrightarrow{E_a}$		
		Check $E_a = f_3(K, N_a, N_b, R_a, IO_a, A, B)$. If check fails, abort.	
	$\xleftarrow{E_b}$		
Check $E_b = f_3(K, N_b, N_a, R_b, IO_b, B, A)$. If check fails, abort.			
At both devices: $LK = f_2(K, N_a, N_b, \text{"}btlk\text{"}, A, B)$			(4)
At both devices: the encryption key K_{enc} is derived based on K and LK. Further messages are encrypted with K_{enc}.			(5)

Fig. 2 Secure Simple Pairing Protocol in PE model usage option (i)

them. Often, the OoB mechanism between two devices is an alternative physical channel like Near Field Communication. See Fig. 4.

2.4 Bluetooth LE Association Models

Bluetooth v4.0 specifies that the Bluetooth LE operational mode has the JustWorks, PE and OoB association models similar to Bluetooth BR/EDR, except that JW and PE do not have use Elliptic Curve Diffie Hellman (DH). This therefore means that no DH key K is jointly established, and so the link key LK computed in stage (4) is not a function of this, but rather only of the other exchanged values. Furthermore, Bluetooth LE does not have the Numeric Comparison (NC) model.

3 Security of the Pairing Protocol

We give a detailed analysis of the security of the Secure Simple Pairing (SSP) protocol for all its models including Bluetooth LE ones, with respect to standard types of attacks on security protocols as defined in Section I.

Note that the KKS, KC and PFS properties are independent of the association model being used. This is because they concern the link key which is computed independent of the association model i.e. NC, PE or OoB.

Initiating device A		Non-initiating device B	
	$\xrightarrow{PK_a}$	$K = SK_b \cdot PK_a \cdot G$	(1)
$K = SK_a \cdot PK_b \cdot G$	$\xleftarrow{PK_b}$		
A selects random R_a and displays as 6-digit number; the user enters the same 6-digit passkey $R_b = R_a$ into B via its device input. Let R_{ai} be the ith bit of R_a for $i = 1, \ldots, 20$. The rest of this stage is performed 20 times for $i = 1, \ldots, 20$.			(2)
Select random N_{ai}. Compute $C_{ai} = f_1(PK_a, PK_b, N_{ai}, R_{ai})$.		Select random N_{bi}. Compute $C_{bi} = f_1(PK_b, PK_a, N_{bi}, R_{bi})$.	
	$\xrightarrow{C_{ai}}$		
	$\xleftarrow{C_{bi}}$		
	$\xrightarrow{N_{ai}}$		
		Check $C_{ai} = f_1(PK_a, PK_b, N_{ai}, R_{bi})$. If check fails, abort.	
	$\xleftarrow{N_{bi}}$		
Check $C_{bi} = f_1(PK_b, PK_a, N_{bi}, R_{ai})$. If check fails, abort.			
Set $N_a = N_{a20}$. Compute $E_a = f_3(K, N_a, N_b, R_a, IO_a, A, B)$.		Set $N_b = N_{b20}$. Compute $E_b = f_3(K, N_b, N_a, R_b, IO_b, B, A)$.	(3)
	$\xrightarrow{E_a}$		
		Check $E_a = f_3(K, N_a, N_b, R_a, IO_a, A, B)$. If check fails, abort.	
	$\xleftarrow{E_b}$		
Check $E_b = f_3(K, N_b, N_a, R_b, IO_b, B, A)$. If check fails, abort.			
At both devices: $LK = f_2(K, N_a, N_b, \text{“}btlk\text{”}, A, B)$			(4)
At both devices: the encryption key K_{enc} is derived based on K and LK. Further messages are encrypted with K_{enc}.			(5)

Fig. 3 Secure Simple Pairing Protocol in PE model usage option (ii)

3.1 Known Key Security (KKS) and Key Control (KC)

The SSP for Bluetooth v4.0 BR/EDR achieves KKS because compromise of any link key of a session does not allow an attacker to compute link keys of other sessions. To see this, recall that

$$LK = f_2(K, N_a, N_b, \text{“}btlk\text{”}, A, B)$$

where f_2 is a non-invertible HMAC construction based on SHA-256. The same KKS security is true for the Bluetooth LE's SSP as it uses the same function f_2.

SSP for both Bluetooth BR/EDR and Bluetooth LE achieve KC because the link key is derived as the output of a cryptographic function f_3 designed to be a secure in the sense of generating random unbiased keys, and for random inputs N_a and N_b (as well as the Diffie-Hellman key K computed in stage (1) for the case of Bluetooth BR/EDR) from equally contributed parts from A and B.

3.2 Perfect Forward Secrecy

For a DH-based key exchange protocol where ephemeral private exponents are used, then it provides perfect forward secrecy (PFS) [20]. The basic idea for achieving PFS is not to have long-term private keys that are never updated, but rather to be generated afresh with some frequency. In the case of the SSP of Bluetooth BR/EDR irrespective

Initiating device A		Non-initiating device B	
	$\xrightarrow{PK_a}$	$K = SK_b \cdot PK_a \cdot G$	(1)
$K = SK_a \cdot PK_b \cdot G$	$\xleftarrow{PK_b}$		
Select random R_a.		Select random R_b.	(2)
Compute $C_a = f_1(PK_a, PK_a, R_a, 0)$.		Compute $C_b = f_1(PK_b, PK_b, R_b, 0)$.	
	$\xRightarrow{A, R_a, C_a}$		
	$\xLeftarrow{B, R_b, C_b}$		
Check $C_b = f_1(PK_b, PK_b, R_b, 0)$.		Check $C_a = f_1(PK_a, PK_a, R_a, 0)$.	
If check fails, abort.		If check fails, abort.	
Select random N_a.		Select random N_b.	
	$\xrightarrow{N_a}$		
	$\xleftarrow{N_b}$		
Compute $E_a = f_3(K, N_a, N_b, R_a, IO_a, A, B)$.		Compute $E_b = f_3(K, N_b, N_a, R_b, IO_b, B, A)$.	
	$\xrightarrow{E_a}$		
		Check $E_a = f_3(K, N_a, N_b, R_a, IO_a, A, B)$.	
		If check fails, abort.	
	$\xleftarrow{E_b}$		
Check $E_b = f_3(K, N_b, N_a, R_b, IO_b, B, A)$.			
If check fails, abort.			
At both devices: $LK = f_2(K, N_a, N_b, \text{“}btlk\text{”}, A, B)$			(4)
At both devices: the encryption key K_{enc} is derived based on K and LK. Further messages are encrypted with K_{enc}.			(5)

Fig. 4 Secure Simple Pairing Protocol in OoB model, where $\Longrightarrow$ and $\Longleftarrow$ denote OoB communication

of its model, once the long-term private key is compromised then the DH key K is easily computed. Since other parameters used in computing the link key LK are public, this means any link key of a previous session can be computed. It is mentioned in [3] that a public-private key-pair needs to be only generated once, and that it is up to the device if it wants to refresh its private key at all. Thus, PFS for SSP is more of a policy issue that is dependent on the device's decision on its private key update policy. If PFS is a concern, then we recommend to update the private key regularly, or to have the DH key be generated as a function of additional ephemeral random values.

Meanwhile for Bluetooth LE, which does not use elliptic curve Diffie-Hellman (thus no DH key K is generated), the link key is computed using the same function f_2 with values exchanged between A and B e.g. N_a, N_b. While there is no explicit mention in the Bluetooth v4.0 core specification [3] of what shared secret between A and B is input to f_2, it is clearly not a DH key. Yet a shared secret is necessary as input in this Bluetooth LE case otherwise the derived link key will be publicly computable in absence of the DH key. In terms of PFS, what can be said is that Bluetooth LE is more resistant because compromise of the long-term private key does not make it easier to compute the shared secret, in contrast to Bluetooth BR/EDR where the shared secret is the DH key which is immediately computable once the long-term private key is compromised.

3.3 Key Compromise Impersonation

Key compromise impersonation (KCI) [14,4] means that even if the long-term private key of a device e.g. A is compromised, it should not be feasible for an attacker to impersonate any other device B to A. Now for the case of SSP for Bluetooth BR/EDR, when the long-term private key of a device A is compromised, DH key K can be computed. The only obstacle that remains in the way for a KCI attacker is the SAS. Since this SAS is communicated between the pairing devices via an out-of-band channel involving the human user, security then depends on whether the Bluetooth device being impersonated by the KCI adversary is present.

For case I: the user is pairing two legitimate Bluetooth devices A and B that are both owned by him, while in the background the KCI adversary with an adversarial Bluetooth device is trying to impersonate B to A. This is typically so when the user owns both devices, e.g. his mobile phone and his computer. In such a case, KCI resilience can be achieved since the KCI adversary is unable to interfere with the human-aided out-of-band channels that communicate the SAS between A and B. Take the NC model as an example. Note that the user expects to see the same SAS (V_a resp. V_b) displayed on the screen of his two legitimate devices. An adversarial device is unable to interfere with the visual display of the legitimate devices, nor is the adversary able to view what is being displayed. In fact, the same result applies for all other models so for the rest of this section, we will only discuss KCI resilience for case II.

For case II: the user desires to pair his device A with another device B not owned by him; this includes e.g. pairing with Bluetooth-enabled vending machines, in which case B's human owner is clearly not present, and cannot be. In a distributed ad hoc network setting such as Bluetooth with multiple human owners, it is possible for any device to be malicious, hence the need for pairing in the first place. What is thus interesting is if an adversary can use his adversarial device B to attack another legitimate device A. We now discuss how it is possible that a KCI adversary M can violate KCI resilience for this case II.

Consider the NC model. For case II, recall that the SAS V_a (resp. V_b) is a function of public keys as well as nonces, thus a KCI attacker impersonating B can easily choose an arbitrary N_b and thus complete all the steps of an SSP session. In particular, it can compute the V_b that equals the V_a computed by A and thus the same SAS will be viewed at both devices and verified by the user as correct.

For the PE model for option (i), the SAS R_a (resp. R_b) is input by the user at each device, and thus for case II the adversarial device gets the same SAS input from the user as the legitimate device. For the PE model for option (ii), if the adversary is impersonating A to B, then it can simply use any SAS R_a of its choice to be displayed, and the user will input the same into B. If it is impersonating B, it will receive the same SAS input from the user as the SAS displayed to the user by A.

The OoB model is different in the sense that the SAS values R_a and R_b are randomly generated by A and B, respectively, and then communicated to each other via an out-of-band channel, which is assumed to be a physically source-authenticated channel, thus case II would not be exploitable; e.g. a user pairs his device A with another device B for which he has communicated his SAS to the human owner of B. If out-of-band communication is only possible in one direction, then the SAS transmitted this way needs to be kept secret [3]. Thus, if the direction of out-of-band communication is from A to B, then the KCI attacker cannot know R_a and therefore cannot complete the steps of Authentication Stage 2. In fact, even if R_a is non-secret for the case where out-of-band communication is bi-directional, then the KCI attacker still cannot impersonate B since by assumption an out-of-band is a physically source-authenticated channel, thus the out-of-band channel from B to A cannot be impersonated by the attacker. Summarizing, the OoB model achieves KCI resilience.

For Bluetooth LE, similar to our discussion of PFS for Bluetooth LE in the previous subsection, the same can be said here in terms of its KCI resilience. The reason is that KCI considers what security remains in the event that the long-term private key is compromised. For Bluetooth BR/EDR, this compromise means that the shared DH key is immediately computable. In contrast, for Bluetooth LE, this compromise does not eliminate the security of the shared secret because the shared secret is independent of the long-term private key, and therefore provides an extra factor of security. Thus Bluetooth LE achieves KCI resilience.

3.4 MitM Attacks

A man-in-the-middle (MitM) attack is an attack where an adversary M places himself between two (or more) legitimate devices A and B and causes at least one of them to believe falsely about the actual execution of the protocol, e.g. a device A believes it is communicating with a device $E \neq B$ (this is also known as unknown key-share attack), or that A receives a message that differs from what B had sent.

Considering MitM attacks on SSP is interesting not only because SSP is designed explicitly to resist this kind of attack but also because an MitM attack allows to exploit the bit-by-bit evaluation of the passkey by devices to leak out the passkey a bit at a time.

The NC Model. Here, if both legitimate devices A and B are owned by the same human, then the SSP is secure against MitM attack even if an attacker M mounts a key substitution attack [20,19,2,1] by replacing the public key PK_a (resp. PK_b) of device A (resp. B) with its own public key PK_m. This is because B definitely uses PK_b instead of PK_m yet since the SAS V_a (resp. V_b) is a function of the public keys of both devices so a user checking the displayed V_a and V_b on both his devices respectively would find that they are different. Now consider the setting as per case II in section 3.3 where the human owner of device A wants to pair with another device B that is not owned by him. This can be a common case since a Bluetooth device is capable of communicating with several other devices within a wireless piconet, and any device is potentially malicious. In fact, this warrants the need for pairing in the first place. Then the following MitM attack applies (see Fig. 5), where an adversarial device M impersonates B to A, and vice versa can also impersonate A to B:

1. During stage 1, A initiates the pairing protocol with device B by sending its public key PK_a to device B, and expects to receive the public key of B. Nevertheless, M sends to A its own public key PK_m instead.
2. The exchanged DH key is therefore computed by A as
$$K = SK_a \cdot PK_m \cdot G,$$
while M computes it as
$$K = SK_m \cdot PK_a \cdot G.$$
3. During stage 2, M arbitrarily selects a 128-bit nonce N_m and sets $R_m = 0$. Then M computes and sends to A, this value:
$$C_m = f_1(PK_m, PK_a, N_m, R_m),$$

4. A randomly selects a 128-bit nonce N_a and sends this to B which is received by M.
5. M sends N_m to A.
6. A recomputes $f_1(PK_m, PK_a, N_m, R_m)$ and checks if it equals the received C_m. It aborts the protocol if this check does not pass.
7. A computes $V_a = g(PK_a, PK_m, N_a, N_m)$ while M computes $V_m = g(PK_a, PK_m, N_a, N_m)$. Both V_a and V_m will match and thus the attacker can proceed to stage 3 and easily complete the steps since it knows the DH key K shared with A. In the end, M shares a link key

$$LK = f_2(K, N_a, N_m, \text{“}btlk\text{”}, A, B)$$

with A, when A thinks it is paired with B. This additionally achieves the objective of an unknown key-share (UKS) attack [6]. In the same way, an attacker can mount a UKS attack on B.

Initiating device A		Attacker M impersonating responding device B	
	$\xrightarrow{PK_a}$	$K = SK_m \cdot PK_a \cdot G$	(1)
$K = SK_a \cdot PK_m \cdot G$	$\xleftarrow{PK_m}$		
Select random N_a.		Select random N_m.	(2)
Set $R_a = R_m = 0$.		Set $R_m = R_a = 0$.	
		$C_m = f_1(PK_m, PK_a, N_m, R_m)$	
	$\xleftarrow{C_m}$		
	$\xrightarrow{N_a}$		
Check $C_m = f_1(PK_m, PK_a, N_m, R_m)$	$\xleftarrow{N_m}$		
If check fails, abort.			
Compute $V_a = g(PK_a, PK_m, N_a, N_m)$.		Compute $V_m = g(PK_a, PK_m, N_a, N_m)$.	
At each device (A, M), V_a (resp. V_m) is displayed as a 6-digit number.			
Compute $E_a = f_3(K, N_a, N_m, R_a, IO_a, A, B)$.		Compute $E_m = f_3(K, N_m, N_a, R_m, IO_b, B, A)$.	(3)
	$\xrightarrow{E_a}$		
	$\xleftarrow{E_m}$		
Check $E_m = f_3(K, N_m, N_a, R_m, IO_b, B, A)$.			
If check fails, abort.			
At both devices: $LK = f_2(K, N_a, N_m, \text{“}btlk\text{”}, A, B)$			(4)
At both devices: the encryption key K_{enc} is derived based on K and LK. Further messages are encrypted with K_{enc}.			(5)

Fig. 5 MitM attack against SSP in the NC model for case II

The PE(i) Model. We first describe dictionary-style attacks on the PE model for its usage option (i) that apply even if the impersonated device is present. This attack carries through all five stages of Bluetooth's SSP. To motivate, we start with a naive attack. See Fig. 6 for an illustration. Note that a double arrow like $\xleftarrow{PK_m}\xleftarrow{PK_b}$ means that the message containing PK_b is intercepted by the attacker and replaced by one containing PK_m. The attack is as follows, where the malicious MitM attacker is denoted as M.

Initiating device A	M	Non-initiating device B	
	$\xrightarrow{PK_a}\xrightarrow{PK_m}$	$K_2 = SK_b \cdot PK_m \cdot G$	(1)
$K_1 = SK_a \cdot PK_m \cdot G$	$\xleftarrow{PK_m}\xleftarrow{PK_b}$		
	$K_1 = SK_m \cdot PK_a \cdot G$		
	$K_2 = SK_m \cdot PK_b \cdot G$		
	At each device (A, B), the user enters the same 6-digit passkey $R_a = R_b$ via the device input.		(2)
	Let R_{ai} be the ith bit of R_a for $i = 1, \ldots, 20$. The rest of this stage is performed up to 20 times for $i = 1, \ldots, 20$.		
	If the protocol aborts prematurely, the user tries once more by inputting the same passkey into devices A and B.		
Select random N_{ai}.	For iteration i, M guesses a bit R_{mi} and selects a random N_{mi}.	Select random N_{bi}.	
$C_{ai} = f_1(PK_a, PK_m, N_{ai}, R_{ai})$	$C_{mai} = f_1(PK_m, PK_b, N_{mi}, R_{mi})$	$C_{bi} = f_1(PK_b, PK_m, N_{bi}, R_{bi})$	
	$C_{mbi} = f_1(PK_m, PK_a, N_{mi}, R_{mi})$		
	$\xrightarrow{C_{ai}}\xrightarrow{C_{mai}}$		
	$\xleftarrow{C_{mbi}}\xleftarrow{C_{bi}}$		
	$\xrightarrow{N_{ai}}\xrightarrow{N_{mi}}$	Check $C_{mai} = f_1(PK_m, PK_b, N_{mi}, R_{bi})$	
Check $C_{mbi} = f_1(PK_m, PK_a, N_{mi}, R_{ai})$	$\xleftarrow{N_{mi}}\xleftarrow{N_{bi}}$	If check fails, abort.	
If check fails, abort.			
$N_a = N_{a20}$	$N_m = N_{m20}$	$N_b = N_{b20}$	(3)
$E_a = f_3(K_1, N_a, N_m, R_a, IO_a, A, B)$	$E_{ma} = f_3(K_2, N_m, N_b, R_a, IO_a, A, B)$	$E_b = f_3(K_2, N_b, N_m, R_b, IO_b, B, A)$	
	$E_{mb} = f_3(K_1, N_m, N_a, R_b, IO_b, B, A)$		
	$\xrightarrow{E_a}\xrightarrow{E_{ma}}$	Check $E_b = f_3(K_1, N_m, N_a, R_b, IO_b, B, A)$	
Check $E_b = f_3(K_1, N_m, N_a, R_b, IO_b, B, A)$	$\xleftarrow{E_{mb}}\xleftarrow{E_b}$	If check fails, abort.	
If check fails, abort.			
	At both devices: $LK_{am} = f_2(K_1, N_a, N_b,$ "*btlk*"$, A, B)$		(4)
	$LK_{bm} = f_2(K_2, N_a, N_b,$ "*btlk*"$, A, B)$		
	At both devices: the encryption keys are derived based on K_1, K_2, LK_{am} and LK_{bm}, respectively.		(5)
	Further messages are encrypted with the encryption keys.		

Fig. 6 Naive MitM attack against PE(i) model

1. During stage 1, A initiates the pairing protocol with device B by sending its public key PK_a to device B. However, M performs a basic public key substitution attack [20,19,2,1] by replacing PK_a with its own public key PK_m.
2. B receives PK_m thinking it is A's public key. It responds with its own public key PK_b to A. But similarly, M replaces PK_b with its public key PK_m. Thus A receives PK_m thinking it is the public key of B.
3. The exchanged DH key is therefore computed by A as

$$K_1 = SK_a \cdot PK_m \cdot G,$$

while the DH key computed by B is

$$K_2 = SK_b \cdot PK_m \cdot G.$$

M can compute both keys as

$$K_1 = SK_m \cdot PK_a \cdot G,$$

$$K_2 = SK_m \cdot PK_b \cdot G.$$

4. During stage 2, for $i = 1, \ldots, 20$, then A will randomly select a 128-bit nonce N_{ai}, and note that R_{ai} is a bit from the R_a input by the user. A computes

$$C_{ai} = f_1(PK_a, PK_m, N_{ai}, R_{ai}).$$

C_{ai} is sent to B, but this is replaced by M with C_{mai} computed as:

$$C_{mai} = f_1(PK_m, PK_b, N_{mi}, R_{mi}), \tag{1}$$

where N_{mi} and R_{mi} are values randomly chosen by M. Note that R_{mi} is in fact M's guess of the bit $R_{ai} = R_{bi}$.

5. Thus B receives C_{mai} and randomly selects a 128-bit nonce N_{bi}, and similarly R_{bi} is the corresponding bit of $R_b = R_a$ input by the user. B computes

$$C_{bi} = f_1(PK_b, PK_m, N_{bi}, R_{bi}),$$

and sends C_{bi} is sent to A. This is replaced by M with C_{mbi} computed as:

$$C_{mbi} = f_1(PK_m, PK_a, N_{mi}, R_{mi}). \tag{2}$$

6. Upon receiving C_{mbi}, A sends N_{ai} to B, but this is replaced by M with N_{mi}.
7. B recomputes $f_1(PK_m, PK_b, N_{mi}, R_{bi})$ and checks if it equals the received C_{mai}. It aborts the protocol if this check does not pass. Note that the C_{mai} sent by M to B computed as in equation (1) will equal the value computed by B if R_{mi} equals R_{bi}. This will occur with probability $\frac{1}{2}$. In the case that the check does not pass and the session is aborted, and a new session is restarted, it is highly likely that the human user will input the same passkey $R_a = R_b$. Furthermore, from the previous abort, the attacker clearly knows that his guess of R_{bi} was wrong, thus it must be that $R_{bi} = \overline{R_{mi}}$. This allows him to now bypass R_{bi} easily and proceed to guessing the next bit of R_b.
8. If the check passes, B sends N_{bi} to A but this is replaced by M with N_{mi}.
9. Upon receiving N_{mi}, A recomputes $f_1(PK_m, PK_a, N_{mi}, R_{ai})$ and checks if it equals the received C_{mbi}. It aborts the protocol if this check does not pass. Note that the C_{mbi} sent by M to A computed as in equation (2) will equal the value computed by A if R_{mi} equals R_{ai}. Since $R_{ai} = R_{bi}$, this check will pass if it previously passed at B for C_{mai}.
10. Eventually M obtains the bits of $R_{ai} = R_{bi}$ (for $i = 1, \ldots, 20$). Now he can proceed to stage 3 and easily complete the steps as illustrated in Fig. 6 since it knows the keys K_1 and K_2 shared with A and B, and also the 20-bit passkey $R_a = R_b$. In the end, M shares a link key with A, when A thinks it is paired with B; and M also shares a link key with B when B thinks it is paired with A. This violates the resistance of SSP against MitM attacks, and additionally achieves the objective of an unknown key-share (UKS) attack [6].

The number of bits the attacker can gain in one session (before a premature abort) is computed as follows. The attacker will always be able to verify at least the first guessed bit R_{a1}. So he can gain in one protocol run at least one bit and at most all 20 bits. The probability for him to get exactly one bit is the same as the probability of a wrong guess at first try, which is $\frac{1}{2}$. Then, the probability for him to get exactly two bits is the same as a success for the first try and a wrong guess for the second try, which is $\frac{1}{4}$. Enumerating this, we obtain more generally the probability for the attacker to retrieve exactly n bits in one run is given by $\frac{1}{2^n}$. Thus, the average number of bits an attacker can expect to get in one run is:

$$\sum_{n=1}^{20} n \cdot \frac{1}{2^n}$$

which tends toward 2. So with just 10 tries, s/he can obtain the entire 20-bit passkey. In contrast, any 20-bit passkey should have a probability of 2^{-20} of being guessed correct, which requires on average 2^{19} guesses before the correct match is obtained. Yet, this naive attack is more to make a theoretical point: that the passkey is not meant to be really secure against offline dictionary attacks, but rather that a passkey in the context of SSP is more a SAS than a long-term human-memorable secret. So protection against offline dictionary attacks is not vital since a user would generally not reuse the same passkey indefinitely, but only for the particular short period of time when he is trying to cause a successful pairing.

Our attack above was aided by the fact that the session may abort prematurely before all 20 bits of the passkey are used, as soon as a passkey bit is guessed incorrectly, and this allowed the attacker to precisely verify any particular bit of the passkey. Thus the attack will be made more complicated if there is no premature abort but instead the SSP continues its session till all 20 bits are used even if some bits in between are guessed incorrectly.

In fact, we can do better than the naive attack. We start with the intuition. The underlying idea is that although N_{ai} for C_{ai} is initially withheld, it is eventually revealed within the same stage 2 of the SSP in the Passkey Entry model. Thus, the only unknown parameter used in the computation of C_{ai} is in fact R_{ai} which is just one bit. Hence, an attacker can easily mount an offline dictionary attack by making a guess R'_{ai} of R_{ai} and recomputing $f_1(PK_a, PK_b, N_{ai}, R'_{ai})$ to check if it equals C_{ai}. This requires only one guess R'_{ai} of R_{ai}. In the same way, repeating this for all $i = 1 \ldots 20$, he needs only $20 \approx 2^{4.5}$ guesses to recover the entire 20 bits of R_a by offline dictionary attack. In contrast, a 20-bit passkey should require 2^{19} guesses on average. We note that this observation was independently made en passant in [16]. However, this observation is not complete in itself because the SSP by design does not allow the attacker any advantage even if he obtains the passkey. This is because without knowledge of the DH key K shared between devices A and B, an attacker cannot pass the next stage 3 and thus cannot compute any link key. Nevertheless, we now show how to mount a full attack through all stages of the SSP, so that the attacker can in the end share a link key with both devices.

The idea in the Passkey Entry model of SSP is to complicate offline dictionary attacks by treating in sequence each bit independently rather than all 20 bits in one instance. To quote [3]: "The gradual disclosure prevents leakage of more than 1 bit of un-guessed Passkey information in the case of a man-in-the-middle attack." Yet less disclosure means dependence on less secret bits, thus less bits to guess for the attacker, and so a higher probability that the guess is correct. This is a double-edged sword. Our attack shows that this "gradual disclosure" in fact causes it to be much more insecure than a naive design that treats the 20 bits in one instance.

1. M eavesdrops on an SSP protocol session between A and B, and obtains the values C_{ai} and R_{ai} (for $i = 1 \ldots 20$), or the values C_{bi} and R_{bi} (for $i = 1 \ldots 20$), and then flips any bit of any message in stage 3 causing the session to abort.
2. M then mounts an offline dictionary attack as described in the paragraphs above, to recover $R_a = R_b$.
3. A re-initiates a new SSP protocol session with device B by sending its public key PK_a to device B. However, M performs a public key substitution attack [20,19,2] by replacing PK_a with its own public key PK_m.

4. B receives PK_m thinking it is A's public key. It responds with its own public key PK_b to A. But similarly, M replaces PK_b with its public key PK_m. Thus A receives PK_m thinking it is the public key of B.
5. Thus the DH key computed by A is
$$K_1 = SK_a \cdot PK_m \cdot G,$$
while the DH key computed by B is
$$K_2 = SK_b \cdot PK_m \cdot G.$$
M can compute both keys as
$$K_1 = SK_m \cdot PK_a \cdot G,$$
$$K_2 = SK_m \cdot PK_b \cdot G.$$
6. During stage 2, the user re-inputs the same passkey, $R_a = R_b$ as during the previous aborted session.
7. Since M has obtained the bits of $R_{ai} = R_{bi}$ (for $i = 1 \ldots 20$) by mounting an offline dictionary attack, he can now proceed to stage 3 and easily complete the steps as illustrated in Fig. 6 since it knows the keys K_1 and K_2 shared with A and B, and also the 20-bit passkey $R_a = R_b$. In the end, M shares a link key with A with A thinking it is paired with B; and M also shares a link key with B with B thinking it is paired with A.

This dictionary-style attacks exploit the fact that a human user typically re-enters the same passkey after a few error messages. This is so since humans can potentially be influenced by their common interaction with PINs for automatic teller machines or for their mobile phones, which are long-term PINs in contrast to the one-off ephemeral SAS passkey. The first few times the machine outputs error messages usually leads the human to believe it is due to mechanical limitations of the input keypad e.g. he had mistyped a digit, or his press on a digit had either been too soft that the digit press was not detected by the device, or too hard that the digit press was detected by the devices as more than one press of the same digit.

The second attack shows that the SAS (passkey) should never be reused, even for once, and even if a session aborted pre-maturely. Instead, the user should not even assume any mechanical errors had occurred for premature abortions but rather start afresh with a totally different passkey.

Note that dictionary-style attacks only apply for PE model usage option (i) since the SAS passkey is then a short string entered by the human user, while for PE model usage option (ii) the SAS passkey is actually randomly generated by the initiating device.

The PE (ii) Model. We now show an MitM attack for the PE model usage option (ii), for the setting as per case II of Section 3.3, i.e. where a legitimate device B is present, and the attacker M is impersonating A which is not present. We exploit the fact that for this case, the attacker M can generate and display any passkey of its choice. See Fig. 7. The attack follows:

1. During stage 1, M impersonating A initiates the pairing protocol with device B by sending its public key PK_m to device B.

M impersonating initiating device A		Non-initiating device B	
	$\xrightarrow{PK_m}$	$K = SK_b \cdot PK_m \cdot G$	(1)
	$\xleftarrow{PK_b}$		
$K = SK_m \cdot PK_b \cdot G$			
M selects a random passkey R_m and displays it as a 6-digit number. The user enters the same 6-digit passkey $R_b = R_m$ into B via its device input. Let R_{mi} be the ith bit of R_m for $i = 1, \ldots, 20$. The rest of this stage is performed up to 20 times for $i = 1, \ldots, 20$.			(2)
Select random N_{mi}. $C_{mai} = f_1(PK_m, PK_b, N_{mi}, R_{mi})$		Select random N_{bi}. $C_{bi} = f_1(PK_b, PK_m, N_{bi}, R_{bi})$	
	$\xrightarrow{C_{mai}}$		
	$\xleftarrow{C_{bi}}$		
	$\xrightarrow{N_{mi}}$	Check $C_{mai} = f_1(PK_m, PK_b, N_{mi}, R_{bi})$ If check fails, abort.	
	$\xleftarrow{N_{bi}}$		
$N_b = N_{b20}$ $E_{ma} = f_3(K, N_m, N_b, R_a, IO_a, A, B)$		$E_b = f_3(K, N_b, N_m, R_b, IO_b, B, A)$	(3)
	$\xrightarrow{E_{ma}}$		
		Check $E_{ma} = f_3(K, N_m, N_b, R_a, IO_a, A, B)$ If check fails, abort.	
	$\xleftarrow{E_b}$		
At both devices: $LK = f_2(K, N_m, N_b, \text{“}btlk\text{”}, A, B)$			(4)
At both devices: the encryption keys are derived based on K and LK. Further messages are encrypted with the encryption keys.			(5)

Fig. 7 MitM attack against PE(ii) model for case II

2. The exchanged DH key is therefore computed by B as

$$K = SK_b \cdot PK_m \cdot G,$$

while M computes it as

$$K = SK_m \cdot PK_b \cdot G.$$

3. During stage 2, M arbitrarily selects a passkey R_m and displays this as a 6-digit number on its screen. The user enters the same 6-digit passkey $R_b = R_m$ into B via its device input.
4. Let R_{mi} be the ith bit of R_m; then repeating for $i = 1, \ldots, 20$, the following steps are performed: M selects an arbitrary nonce N_{mi}; and computes and sends this to B:

$$C_{mai} = f_1(PK_m, PK_b, N_{mi}, R_{mi}),$$

5. B randomly selects a 128-bit nonce N_{bi} and computes

$$C_{bi} = f_1(PK_b, PK_m, N_{bi}, R_{bi}),$$

and sends this to A which is received by M.
6. M sends N_{mi} to B.
7. B recomputes $f_1(PK_m, PK_b, N_{mi}, R_{bi})$ and checks if it equals the received C_{mai}. It aborts the protocol if this check does not pass. It then sends N_{bi} to A which is received by M.

8. The attack proceeds to stage 3 and is easily completed since M knows the DH key K shared with B. In the end, M shares a link key

$$LK = f_2(K, N_m, N_b, \text{“}btlk\text{”}, A, B)$$

with B, when B thinks it is paired with A. This additionally achieves the objective of an unknown key-share (UKS) attack [6].

In fact, a similar attack applies for the reverse direction, i.e. a legitimate device A is present, while the attacker M is impersonating B which is not present. In this case, the adversarial device of M will obtain the input passkey entered by the user.

The OoB Model. The above MiTM attacks do not apply to the OoB model, assuming that the adversary has no access to the OoB channel, since this would otherwise be outside the scope of protocol analysis.

Bluetooth LE Models. The MitM attacks described in this subsection do not apply to the SSP of Bluetooth LE. The reason is because LE differs from Bluetooth BR/EDR only in the non-usage of the DH key as input to the link key computation function f_2. While the MitM attack allows the adversary to influence the DH key via its public key, such an attack does not allow control nor knowledge of the shared secret between A and B used in place of the DH key. Therefore, an MitM attack would not be able to proceed past stage (3).

4 Concluding Remarks

We have presented a detailed security analysis of Bluetooth v4.0's Secure Simple Pairing (SSP) protocol in both BR/EDR and LE operational modes, for all its association models that aim to provide MitM security, i.e. NC, PE(i), PE(ii) and OoB.

It is prudent that a revised version of a protocol, such as is the Bluetooth v4.0, undergo a continual security analysis process by designers and third parties. In this paper, our intention is to contribute to this process.

Table 3 Security Properties of SSP Association Models for Bluetooth BR/EDR and LE

	OoB	NC	PE(ii)	PE(i)		OoB	PE(ii)	PE(i)
KKS	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$
KC	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$	$\surd$	$\surd$
PFS	$\star$	$\star$	$\star$	$\star$		$\surd$	$\surd$	$\surd$
KCI	$\surd$	$\surd^{I},\times^{II}$	$\surd^{I},\times^{II}$	$\surd^{I},\times^{II}$		$\surd$	$\surd$	$\surd$
UKS/MitM	$\surd^{*}$	$\surd^{I},\times^{II}$	$\surd^{I},\times^{II}$	$\times$		$\surd$	$\surd$	$\surd$

Table 3 summarizes our results, sorted from left to right based on the order of priority for SSP models specified by Bluetooth v4.0, according to the IO capability of a device. Achievement of a security property is denoted by a $\surd$, non-achievement by an $\times$. The first four columns denote the SSP models for Bluetooth BR/EDR while the last three columns denote the SSP models for Bluetooth LE. The symbol $\star$ denotes that the security is policy-dependent, while the superscripts I,II refer to cases I, II as described in Section III.C. The superscript * denotes that security is dependent on the OoB channel e.g. Near Field Communication or human-involved channel, whose security consideration is external to typical protocol analysis.

These results indicate that the priority order of these models has been well chosen by the Bluetooth v4.0 specification, i.e. OoB offers the best security if the external OoB channel is assumed secure. That said, this has to be taken with a grain of salt. The gist is the assumption that the underlying OoB channel is secure. This depends on the kind of OoB channel; for instance, Kainda et al. [15] show that human-involved OoB channels should be used carefully by taking into consideration human-involved factors, e.g. non-compelling OoB channels may be susceptible to humans comparing passkeys inaccurately due to distraction or bypassing some steps due to laziness. In the absence of a secure OoB channel, NC model is the next alternative, or else PE(ii) model should be used, and where the PE(i) model is really seldom used. Thinking further on the latter point, the fact that the PE(i) model has been included in the specification, apparently indicates that there could be cases (even if it is hard to think of real concrete examples) where the PE(ii) model cannot be used for which the PE(i) model would be; this would be a fair assumption. Having said that, specifications like SSP should be careful not to include models that exhibit vulnerabilities, even if these vulnerabilities can be avoided by choosing other models. Indeed, if this is the case, is it useful in practice to still include the weaker model under the list of MitM-secure models?

Summarizing the table, we can say that while the most appropriate model in the performance sense is based on the IO capability of the device, in contrast in the security sense the question of which is the best model depends on the underlying assumptions. For instance, OoB is the best choice if the underlying OoB channel is secure. NC can be used if no secure OoB channel exists other than a human being able to view the screen and to press a 'Yes/No' button at each device, as long as the human factors (human mistakes, human laziness) are taken into account during the human interface design. Otherwise, it may make more sense to employ the PE model to compel the human to explicitly type in the passkey digits leaving less margin for comparison error.

Despite that the SSP for Bluetooth BR/EDR was designed with multi-factor authentication layers, namely the use of elliptic curve public-private keys to protect the secrecy of the established DH key, and the use of out-of-band SAS to provide authentication of communicated protocol message transcripts without needing a PKI; however, as we highlighted, the fact that key substitution attacks are possible on the DH key exchange of Bluetooth BR/EDR's SSP leads to cases where an adversary computes the shared DH key and subsequently has less obstacles in bypassing the security mechanisms within the SSP, i.e. he has one less authentication factor to worry about. In contrast, the Bluetooth LE does not exhibit this problem since the DH key is not used. Furthermore, the non-usage of the DH key in Bluetooth LE also means that compromise of the long-term private key does not affect the shared secret between Bluetooth parties. Indeed, the gist of the PFS, KCI and MitM attacks on Bluetooth BR/EDR exploit the fact that the private key directly influences the shared DH key; hence Bluetooth LE is not affected by PFS, KCI or MitM attacks.

In order to address the key substitution attack issue, we need to revisit the fundamental problem behind it, i.e. the lack of initial trust setup. An example that allows for setting up initial trust is the public key infrastructure (PKI) leveraging on the exchange of certificates, which allow parties to have trust in the correct long-term public key of other parties. In contrast, Bluetooth allows devices during stage 1 of an SSP protocol run, to send their public keys to each other (such public keys could be generated in advance or a device could choose to generate a new one after discarding an existing one); at the end of stage 1, a common DH key is generated based on the public key received from the other device. All this is done without any authentication [6], thus making key substitution possible.

It is worthwhile to mention here the relation between Bluetooth's SSP and the wUSB standard [27] to see the difference in their resistance to key substitution. The wUSB protocol has similar association models i.e. a connected model and a numeric comparison model. Also, wUSB is designed such that key substitution attacks cannot work because exchange of public keys between devices (this would correspond to SSP stage 1) is part of a ceremony [27,7] that involves the human user verifying via an out-of-band mechanism that the public keys correspond to correct public IDs of the devices. This completes the initial trust setup, i.e. trust in each other device's public key. Thus, to overcome the key substitution issue, we suggest that for SSP similarly to have its stage (1) be via an out-of-band authenticated channel involving the human user to setup the initial trust in each device's public key, so that key substitution attacks can be prevented, and as a consequence, all the UKS and MitM security issues we highlighted here can be avoided.

Furthermore, we remark that wUSB does not have any problem with respect to PFS since its public-private key-pair is totally ephemeral and needs to be freshly generated for every pairing.

References

1. S.S. Al-Riyami, K.G. Paterson, "Tripartite Authenticated Key Agreement Protocols from Pairings," *Proceedings of IMA Cryptography and Coding '03*, LNCS 2898, pp. 332-359, 2003.
2. S. Blake-Wilson, A. Menezes, "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol," *Proceedings of Public Key Cryptography (PKC '99)*, LNCS 1560, pp. 154-170, 1999.
3. Bluetooth SIG, "Bluetooth Core Specification v4.0," 30 June 2010. Available online at http://bluetooth.com/English/Technology/Building/Pages/Specification.aspx.
4. C. Boyd, A. Mathuria, *Protocols for Authentication and Key Establishment*, Springer-Verlag, 2003.
5. R. Chang, V. Shmatikov, "Formal Analysis of Authentication in Bluetooth Device Pairing," *Proceedings of LICS/ICALP Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA '07)*, July 2007.
6. W. Diffie, P.C. van Oorschot, M.J. Wiener, "Authentication and Authenticated Key Exchange," *Designs, Codes and Cryptography*, Vol. 2, pp. 107-125, 1992.
7. C. Ellison, "Ceremony Design and Analysis," 17 October 2007. Available online at IACR ePrint Archive, http://eprint.iacr.org/2007/399.
8. K. Haataja, P. Toivanen, "Practical Man-in-the-Middle Attacks against Bluetooth Secure Simple Pairing," *Proceedings of IEEE International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08)*, pp. 1-5, 2008.
9. K. Haataja, P. Toivanen, "Two Practical Man-in-the-Middle Attacks on Bluetooth Secure Simple Pairing and Countermeasures," *IEEE Transactions on Wireless Communications*, Vol. 9, No. 1, pp. 384-392, 2010.
10. K. Hyppönen, K. Haataja, "Niño Man-in-the-Middle Attack on Bluetooth Secure Simple Pairing," *Proceedings of IEEE International Conference in Central Asia on Internet (ICI '07)*, pp. 1-5, 2007.
11. K. Hyppönen, K. Haataja, "Man-in-the-Middle Attacks on Bluetooth: a Comparative Analysis, a Novel Attack, and Countermeasures," *Proceedings of IEEE International Symposium on Communications, Coding and Signal Processing (ISCCSP '08)*, pp. 1096-1102, 2008.

12. M.E. Hoque, F. Rahman, S.I. Ahamed, J.H. Park, “Enhancing Privacy and Security of RFID System with Serverless Authentication and Search Protocols in Pervasive Environments,” *Wireless Personal Communications*, Vol. 55, No. 1, pp. 65-79, 2010.
13. B. Howard, “The Bluetooth Car,” PC Magazine, 30 January, 2004.
14. M. Just, S. Vaudenay, “Authenticated Multi-Party Key Agreement,” *Advances in Cryptology - Asiacrypt ’96*, LNCS 1163, pp. 36-49, 1996.
15. R. Kainda, I. Flechais, A.W. Roscoe, “Usability and Security of Out-of-Band Channels in Secure Device Pairing Protocols,” *Proceedings of Symposium on Usable Privacy and Security (SOUPS ’09)*, 2009.
16. C. Kuo, J. Walker, A. Perrig, “Low-cost Manufacturing, Usability, and Security: An Analysis of Bluetooth Simple Pairing and Wi-Fi Protected Setup,” *Proceedings of International Conference on Usable Security (USEC ’07)*, pp. 325-340, 2007.
17. A. Lindell, “Bluetooth v2.1 - a New Security Infrastructure and New Vulnerabilities,” *BlackHat Briefings*, Las Vegas, 2008.
18. A. Lindell, “Attacks on Password Pairing in Bluetooth v2.1,” *CSI ’08*, Maryland, 2008.
19. A. Menezes, M. Qu, S. Vanstone, “Some New Key Agreement Protocols Providing Mutual Implicit Authentication,” *Proceedings of Selected Areas in Cryptography (SAC ’95)*, pp. 22-32, 1995.
20. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
21. P. Mingard, “Elliptic Curve based Diffie-Hellman in Bluetooth v2.1,” Master of Computer Science Semester Project Thesis, EPFL, Switzerland, 2007.
22. C.J. Mitchell, M. Ward, P. Wilson, “Key Control in Key Agreement Protocols," *IEE Electronics Letters*, Vol. 34, No. 10, pp. 980-981, 1998.
23. J.S. Moon, J.H. Park, D.G. Lee, I.-Y. Lee, “Authentication and ID-based Key Management Protocol in Pervasive Environment,” *Wireless Personal Communications*, Vol. 55, No. 1, pp. 91-103, 2010.
24. C. Ntantogian, C. Xenakis, “One-Pass EAP-AKA Authentication in 3G-WLAN Integrated Networks,” *Wireless Personal Communications*, Vol. 48, No. 4, pp. 569-584, 2009.
25. K. Scarfone, J. Padgette, “Guide to Bluetooth Security,” Special Publication 800-121, NIST, September 2008. Available online at http://csrc.nist.gov/publications/nistpubs/800-121/SP800-121.pdf.
26. J. Suomalainen, J. Valkonen, N. Asokan, “Security Associations in Personal Networks: A Comparative Analysis,” *Proceedings of European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS ’07)*, LNCS 4572, pp. 43-57, 2007.
27. USB Implementers Forum (USB-IF), “Wireless USB Specification,” revision 1.0, 12 May 2005.
28. S. Vaudenay, “Secure Communications over Insecure Channels based on Short Authenticated Strings,” *Advances in Cryptology - CRYPTO ’05*, LNCS 3621, pp. 309-326, 2005.
29. E.-J. Yoon, K.-Y. Yoo, S.-S. Yeo, C. Lee, “Robust Deniable Authentication Protocol,” *Wireless Personal Communications*, Vol. 55, No. 1, pp. 81-90, 2010.