2017

Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard V5.0 and its countermeasure

Da-Zhi Sun
University of Wollongong, dzsun@uow.edu.au

Yi Mu
University of Wollongong, ymu@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Publication Details

Sun, D., Mu, Y. & Susilo, W. (2018). Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard V5.0 and its countermeasure. Personal and Ubiquitous Computing, 22 (1), 55-67.

Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard V5.0 and its countermeasure

Abstract
Bluetooth devices are widely employed in the home network systems. It is important to secure the home members' Bluetooth devices, because they always store and transmit personal sensitive information. In the Bluetooth standard, Secure Simple Pairing (SSP) is an essential security mechanism for Bluetooth devices. We examine the security of SSP in the recent Bluetooth standard V5.0. The passkey entry association model in SSP is analyzed under the man-in-the-middle (MITM) attacks. Our contribution is twofold. (1) We demonstrate that the passkey entry association model is vulnerable to the MITM attack, once the host reuses the passkey. (2) An improved passkey entry protocol is therefore designed to fix the reusing passkey defect in the passkey entry association model. The improved passkey entry protocol can be easily adapted to the Bluetooth standard, because it only uses the basic cryptographic components existed in the Bluetooth standard. Our research results are beneficial to the security enhancement of Bluetooth devices in the home network systems.

Disciplines
Engineering | Science and Technology Studies

Publication Details
Sun, D., Mu, Y. & Susilo, W. (2018). Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard V5.0 and its countermeasure. Personal and Ubiquitous Computing, 22 (1), 55-67.

Man-in-the-Middle Attacks on Secure Simple Pairing in Bluetooth Standard V5.0 and Its Countermeasure

Da-Zhi Sun [a,b,c], Yi Mu [c], Willy Susilo [c]

[a] *Tianjin Key Laboratory of Advanced Networking (TANK), School of Computer Science and Technology, Tianjin University, No. 135, Yaguan Road, Tianjin Haihe Education Park, Tianjin 300350, China*

[b] *State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*

[c] *Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia*

Corresponding author. Tel.: +86 22 27406538; fax: +86 22 27406538; ORCID: 0000-0002-5479-549X.

E-mail addresses: sundazhi@tju.edu.cn, dzsun@uow.edu.au (D.-Z. Sun).

Abstract

Bluetooth devices are widely employed in the home network systems. It is important to secure home members' Bluetooth devices, because they always store and transmit personal sensitive information. In the Bluetooth standard, Secure Simple Pairing (SSP) is an essential security mechanism for Bluetooth devices. We examine the security of SSP in the recent Bluetooth standard V5.0. The passkey entry association model in SSP is analyzed under the man-in-the-middle (MITM) attacks. Our contribution is twofold. (1) We demonstrate that the passkey entry association model is vulnerable to the MITM attack, once the host reuses the passkey. (2) An improved passkey entry protocol is therefore designed to fix the reusing passkey defect in the passkey entry association model. The improved passkey entry protocol can be easily adapted to the Bluetooth standard, because it only uses the basic cryptographic components existed in the Bluetooth standard. Our research results are beneficial to the security enhancement of Bluetooth devices in the home network systems.

Keywords *Bluetooth standard, Secure Simple Pairing, passkey entry, man-in-the-middle attack, home network system*

Man-in-the-Middle Attacks on Secure Simple Pairing in Bluetooth Standard V5.0 and Its Countermeasure

Abstract

Bluetooth devices are widely employed in the home network systems. It is important to secure home members' Bluetooth devices, because they always store and transmit personal sensitive information. In the Bluetooth standard, Secure Simple Pairing (SSP) is an essential security mechanism for Bluetooth devices. We examine the security of SSP in the recent Bluetooth standard V5.0. The passkey entry association model in SSP is analyzed under the man-in-the-middle (MITM) attacks. Our contribution is twofold. (1) We demonstrate that the passkey entry association model is vulnerable to the MITM attack, once the host reuses the passkey. (2) An improved passkey entry protocol is therefore designed to fix the reusing passkey defect in the passkey entry association model. The improved passkey entry protocol can be easily adapted to the Bluetooth standard, because it only uses the basic cryptographic components existed in the Bluetooth standard. Our research results are beneficial to the security enhancement of Bluetooth devices in the home network systems.

Keywords *Bluetooth standard, Secure Simple Pairing, passkey entry, man-in-the-middle attack, home network system*

1 Introduction

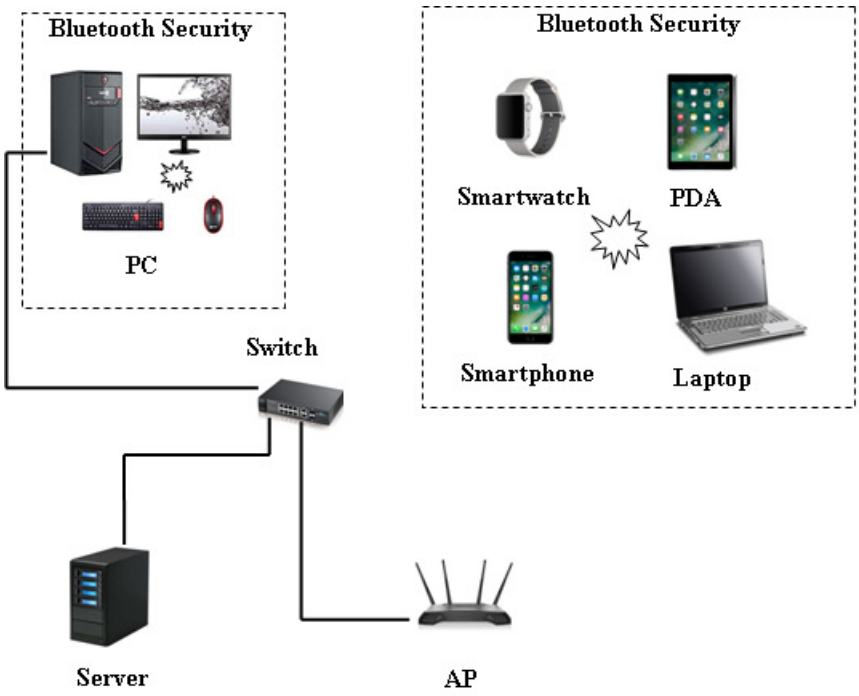

Fig. 1 Bluetooth security for the home network system

The Bluetooth technology [1] enables universal short-range and low-power wireless connectivity among the Bluetooth devices. Nowadays, Bluetooth modules are integrated in most smartphones, wireless headsets, and laptops. It is not surprising that the Bluetooth technology is the footstone of the home network systems.

In practice, the security solutions are necessary to protect the Bluetooth applications [2–8] due to the ubiquitousness of the Bluetooth devices. For a typical home network system, the security overview for Bluetooth devices and their network is described as Fig. 1.

The Bluetooth network is not a traditional IP-based network. Hence, IP-based canonical security solutions, such as IPSec and Public Key Infrastructure (PKI), are not supported by the Bluetooth network. Nevertheless, to guarantee Bluetooth devices and their network in a secure manner, the Bluetooth standard [9, 10] specially provides a set of the exchanging key, authentication, and confidentiality mechanisms.

1.1 Architecture of Bluetooth security

The Bluetooth standard [10] specifies four security modes called security modes 1 through 4. A Bluetooth device possibly supports one or multiple (not all) security modes. Security modes 2 and 4 are the service level enforced security, where security procedures are initiated after physical and logical link setup. Comparatively, security mode 3 is designed for the link level enforced security. That is, a device initiates security procedures, before the physical link is fully established. Security modes 1, 2, and 3 are the legacy security modes and apply to those devices with a controller or a host that does not support security mode 4. Additionally, National Institute of Standards and Technology (NIST) [11] recommends security mode 4, because it requires the secure connections, which use Secure Simple Pairing (SSP) and Advanced Encryption Standard (AES).

From the security view, SSP is the foundation of security mode 4. The crucial task of SSP is to establish the link key between two Bluetooth devices. In fact, the link key dominates the security of the Bluetooth network system, because other security procedures all require depending on the link key to achieve their security goals. During establishing the link key, SSP uses the Elliptic Curve Diffie-Hellman (ECDH) public key algorithm as a means to thwart passive eavesdropping attacks. However, the ECDH public key algorithm in SSP may be subject to man-in-the-middle (MITM) attacks because of the lack of PKI in the Bluetooth network system. To address MITM attacks, the four association models offered in SSP are as follows.

(1) *Numeric comparison* is designed for the case, where a user is shown a 6-digit number on the display of each pairing device and provides a “yes” or “no” response

according to whether the numbers match. Each device in sight of the user independently computes the 6-digit number. Hence, the MITM attacker fails due to the unmatched 6-digit numbers.

(2) *Passkey entry* is primarily designed for the case, where one device has input capability but does not have the capability to display 6 digits and the other device has input and (or) output capabilities. To defeat MITM attacks, a 6-digit number called the passkey need be inputted into one or two devices.

(3) *Out of Band* (OOB) is designed for devices that support an additional wireless, e.g., Near Field Communication (NFC), or wired technology. This model prevents MITM attacks, because it assumes that the attacker cannot compromise two communication channels simultaneously.

(4) *Just works* is designed for the case, where at least one of the pairing devices has neither a display nor a keyboard for entering digits, e.g., headset. This model provides no MITM protection.

1.2 Previous work on Secure Simple Pairing

Chang and Shmatikov [12] used the formal method tool to analyze the numeric comparison association model in SSP and found that the authentication fails if the same device is used parallel in different sessions. Suomalainen et al. [13] pointed out a potential attack scenario, where the security of the device with a more IO (Input and Output) capability is compromised by interacted with another device of restricted IO capability. Lindell [14] proved that the numeric comparison association model in the Bluetooth standard V2.1 is secure under the appropriate security model. Haataja and Toivanen [15, 16] proposed the MITM attacks and the countermeasures for the numeric comparison and OOB association models. The proposed MITM attacks exploit the falsification of information sent during the IO capabilities exchange and the fact that the security of SSP is likely to be limited by the capabilities of the least powerful or the least secure device type. Phan and Mingard [17] mainly analyzed the numeric comparison, passkey entry, and OOB association models using the MITM attacks, providing that one device is malicious. Barnickel et al. [18] explored a MITM attack on the passkey entry association model, when the attacker prevents the pairing process to successfully complete and the user inputs the same passkey twice. Albahar et al. [19–21] presented some countermeasures for SSP to prevent MITM attacks such as using new precautionary steps in the just works association model and building the virtual channel. Gajbhiye et al. [22] presented the simulation and the security analysis of the numeric comparison association model in the network simulator NS2.

1.3 Our contribution

We examine the security of SSP in security mode 4 in the recent Bluetooth standard specifications, i.e., the Bluetooth standard V5.0 [10]. The passkey entry association model in SSP is reevaluated under the MITM attacks. Our contribution is twofold. (1) We demonstrate that the passkey entry association model is vulnerable to the MITM attack, once the host reuses the passkey. (2) An improved passkey entry protocol is therefore designed to fix the reusing passkey problem under the passkey entry association model. Moreover, the implementation cost of the improved passkey entry protocol only has an insignificant increase, compared with the original protocol.

2 Review of Secure Simple Pairing

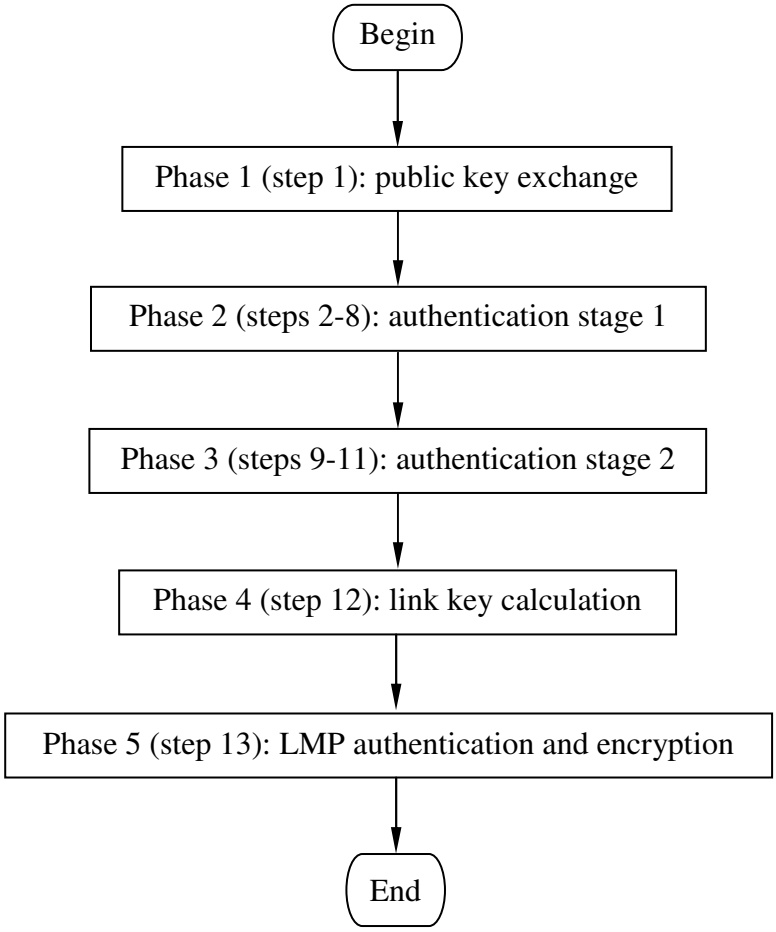

Fig. 2 Flow chart of Secure Simple Pairing

As shown in Fig. 2, SSP consists of 5 phases, i.e., public key exchange (Phase 1), authentication stage 1 (Phase 2), authentication stage 2 (Phase 3), link key calculation (Phase 4), and Link Manager Protocol (LMP) authentication and encryption (Phase 5). Phases 1, 3, 4, and 5 are the same for all association models. However, authentication stage 1 is different depending on the association model used. Before the description of 5

phases in SSP, we firstly introduce the terminology used throughout this paper. For discussion convenience, we almost abide by the same symbols as in the Bluetooth standard [10].

Cryptographic key

PKx: ECDH public key of Bluetooth device X or attacker X.

SKx: ECDH secret (private) key of Bluetooth device X or attacker X.

LK, LK_1, and LK_2: link keys established by Bluetooth devices.

DHkey, $DHkey_1$, and $DHkey_2$: Diffie-Hellman keys.

Constant parameter

BD_ADDRx: unique address of Bluetooth device X.

X: unique address of Bluetooth device X, when it appears in cryptographic function.

IOcapX: IO capabilities of Bluetooth device X.

btlk: a predefined bit string.

Variable parameter

Nx: nonce (unique random value) from Bluetooth device X or attacker X.

Nxi: ith nonce (unique random value) from Bluetooth device X or attacker X.

rx: random value of Bluetooth device X, i.e., the passkey.

rxi: ith bit of the rx.

r*x: random image of the rx.

r*xi: ith bit of the r*x.

Cxi: ith commitment value from Bluetooth device X.

C′xi: ith counterfeit commitment value of Bluetooth device X from the attacker.

Ex: check value from Bluetooth device X.

E′x: counterfeit check value of Bluetooth device X from the attacker.

Cryptographic hash function

f1(): used to generate and verify the Cxi and the C′xi.

f2(): used to generate the link key and possible other keys from the Diffie-Hellman key.

f3(): used to generate and verify the Ex and the E′x in authentication stage 2.

Figure 3 describes 5 phases of SSP under the passkey entry association model. Each Bluetooth device need generate its own ECDH public-private key pair. Here, the key pair can be generated only once per device and may be computed in advance of the pairing procedure. Alternatively, a device may, at any time, discard its public-private key pair and generate a new one instead. We further explain those phases in the following.

2.1 Phase 1: public key exchange

The public key exchange phase is given as follows.

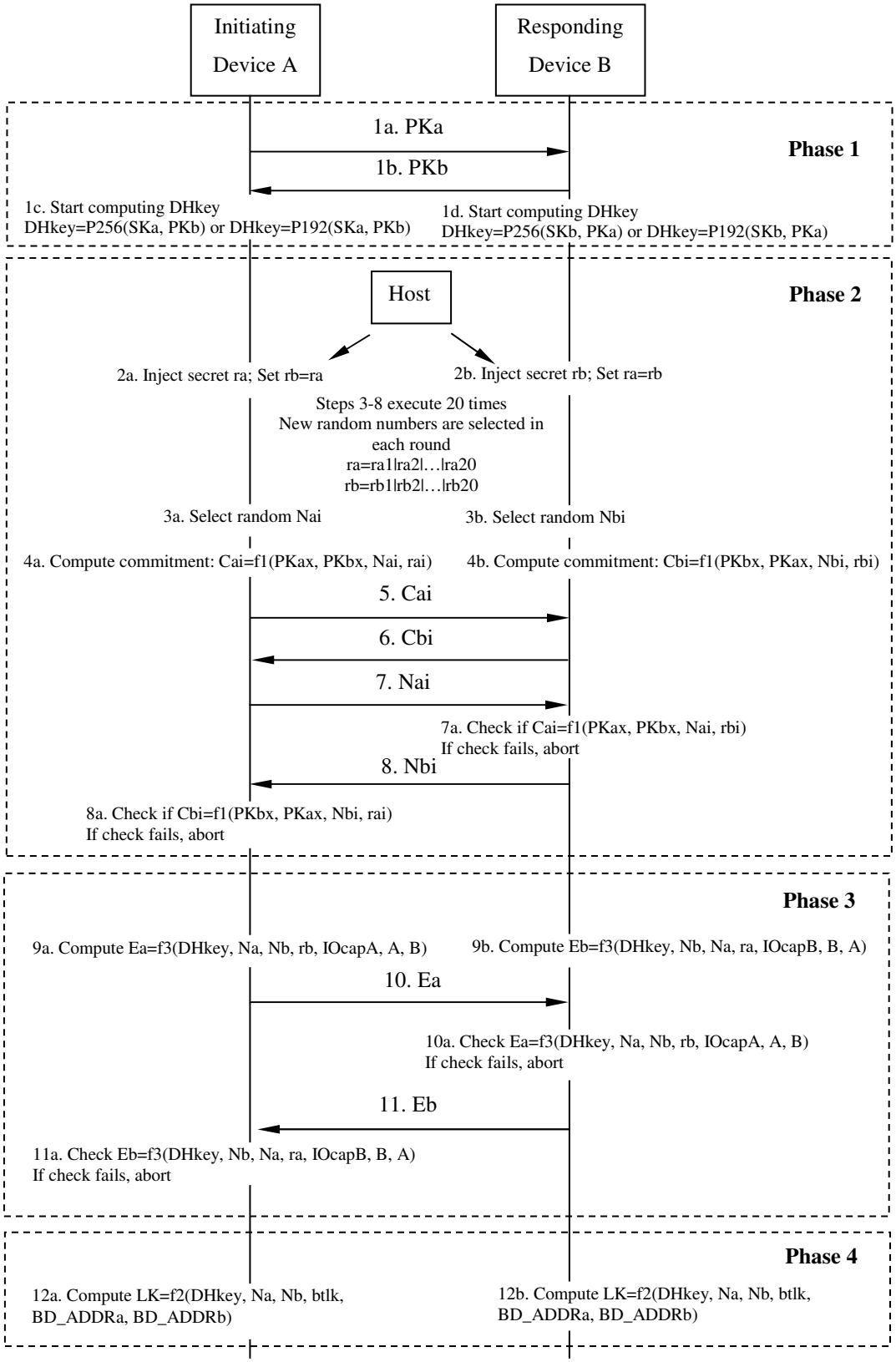

Fig. 3 Secure Simple Pairing under the passkey entry association model

The initiating device A and the responding device B respectively exchange the PKa and the PKb (step 1a and step 1b in Fig. 3). The A computes the shared DHkey by the

Diffie-Hellman function P256(SKa, PKb) or P192(SKa, PKb) (step 1c in Fig. 3). In other side, the B computes the shared DHkey by the Diffie-Hellman function P256(SKb, PKa) or P192(SKb, PKa) (step 1d in Fig. 3). Herein, if both devices' controllers and hosts support secure connections, the function P256() is used. Otherwise, the function P192() is used.

2.2 Phase 2: authentication stage 1

Authentication stage 1 has three different protocols for the corresponding association models, i.e., the numeric comparison protocol, the OOB protocol, and the passkey entry protocol. The just works association model makes use of the numeric comparison protocol. The detail protocol is chosen based on the IO capabilities of the pairing devices. In this section, the passkey entry protocol is reviewed, because we address the weaknesses on it. The host in Fig. 3 is treated as a user or a device function.

The user inputs an identical passkey ra(=rb) into both devices (step 2a and step 2b in Fig. 3). Alternately, the passkey may be generated and displayed on one device, and then the user inputs it into the other (also step 2a and step 2b in Fig. 3). This shared passkey with the 20-bit length is the basis of the mutual authentication of two devices. Steps 3 through 8 are repeated 20 rounds using each bit of the passkey. In the ith round for $1 \leq i \leq 20$, the device A and the device B perform the following steps.

The A and the B respectively generate the random Nai and the random Nbi (step 3a and step 3b in Fig. 3). Then, the A and the B further commit to their ith bit of the passkey by computing Cai=f1(PKax, PKbx, Nai, rai) and Cbi=f1(PKbx, PKax, Nbi, rbi) (step 4a and step 4b in Fig. 3). Here, the PKax and the PKbx respectively denote the x-coordinates of the PKa and the PKb. Then, both devices exchange the Cai and the Cbi (step 5 and step 6 in Fig. 3). Next, the A sends its Nai to the B (step 7 in Fig. 3). Upon receiving the Nai, the B checks whether Cai=f1(PKax, PKbx, Nai, rbi). If it fails, the B terminates the run of the protocol (step 7a in Fig. 3). Otherwise, the B also sends its Nbi to the A (step 8 in Fig. 3). Upon receiving the Nbi, the A checks whether Cbi=f1(PKbx, PKax, Nbi, rai). If it fails, the A terminates the run of protocol (step 8a in Fig. 3).

Note that at the end of this stage, the A and the B all set Na=Na20 and Nb=Nb20 for use in authentication stage 2.

2.3 Phase 3: authentication stage 2

The device A and the device B have successfully completed the message exchange, if they pass the second stage of the authentication. We depict this authentication stage as follows.

The A and the B respectively compute Ea=f3(DHkey, Na, Nb, rb, IOcapA, A, B) and Eb=f3(DHkey, Nb, Na, ra, IOcapB, B, A) as confirmation values (step 9a and step 9b in Fig. 3). The A then transmits its Ea to the B (step 10 in Fig. 3). Upon receiving the Ea, the B checks whether Ea=f3(DHkey, Na, Nb, rb, IOcapA, A, B). If this check fails, it indicates that the A has not confirmed the pairing and the run of the protocol should abort (step 10a in Fig. 3). The B then transmits its Eb to the A (step 11 in Fig. 3). Upon receiving the Eb, the A similarly checks whether Eb=f3(DHkey, Nb, Na, ra, IOcapB, B, A). A failure indicates that the B has not confirmed the pairing and the run of the protocol should abort (step 11a in Fig. 3).

2.4 Phase 4: link key calculation

Once both devices have confirmed the pairing, a shared LK is respectively computed by f2(DHkey, Na, Nb, btlk, BD_ADDRa, BD_ADDRb) (step 12a and step 12b in Fig. 3).

2.5 Phase 5: LMP authentication and encryption

This phase consists of the authentication process and the encryption key generation process, which all base the LK. It is actually the same as the final steps in the pairing of the legacy security modes. The technique details of Phase 5 are omitted, because our research does not focus on them.

3 Vulnerabilities on Secure Simple Pairing

When SSP is under the passkey entry association model, the passkey ra(=rb) is the only secret used to prevent MITM attacks. Clearly, if the attacker knows the ra before the run of the passkey entry protocol, he can launch the MITM attack as Fig. 4. According to the passkey entry protocol, the passkey is generated and displayed by the Bluetooth device or chosen and inputted by the user. According to the Bluetooth standard [10], the passkey generation algorithm is not provided for the device. Hence, the device possibly operates the nonrandom passkey generation algorithm or simply reuses the passkey. The attacker is able to launch the MITM attack in Fig. 4, once he correctly predicts the ra displayed by the device. On the other hand, many users are inclined to choose the same passkey in a period of time, because it is convenient to them. In the following, we depend on this fact to compromise the passkey. Clearly, it leads to the MITM attack on the passkey entry association model as Fig. 4, when the compromised passkey is used again.

3.1 Offline attack on passkey

The attacker E can perform the following steps to derive the used ra.

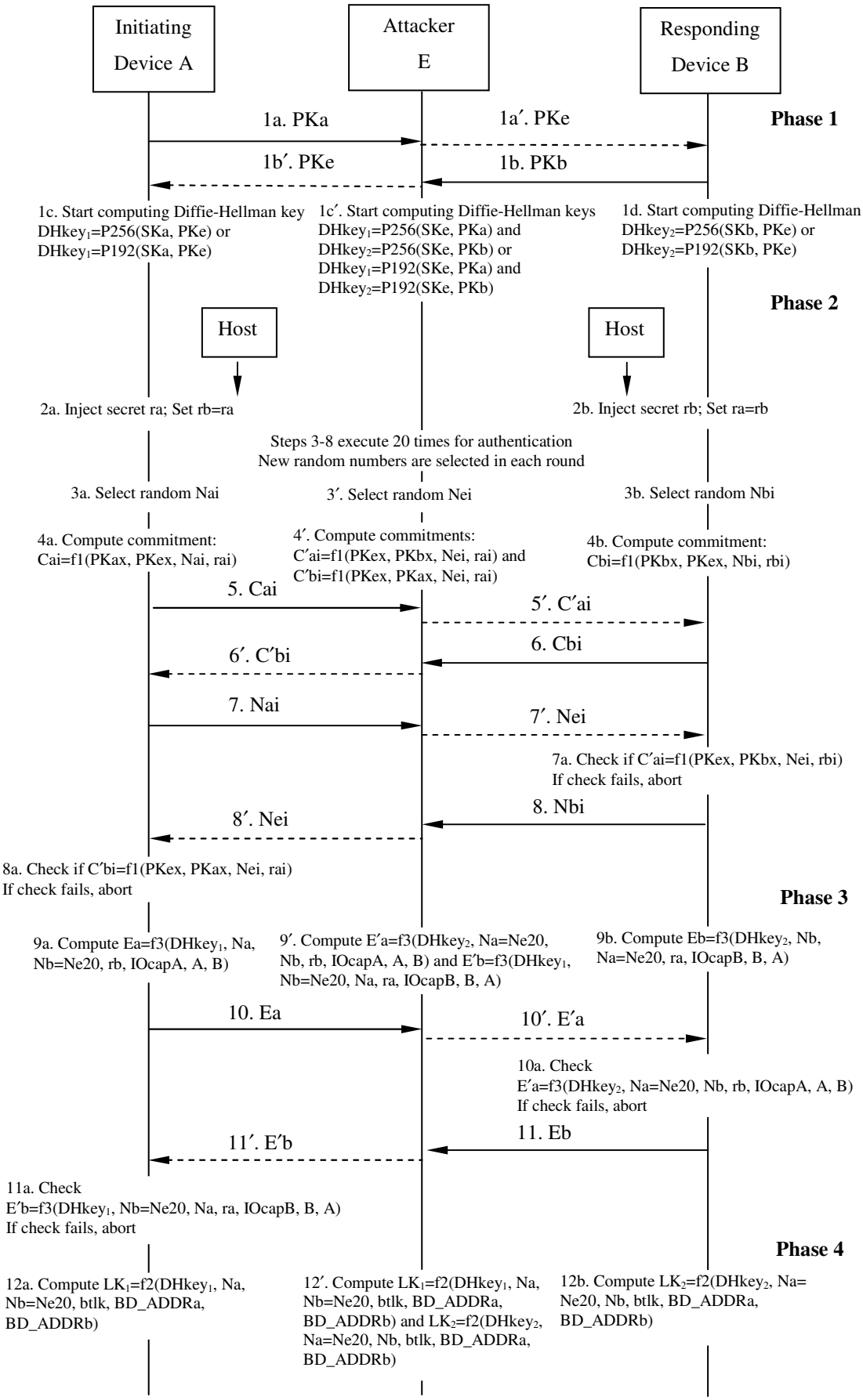

Fig. 4 MITM attack on the passkey entry association model

Step 1. Intercept the PKa and the PKb during the phase of public key exchange (step 1a and step 1b of Fig. 3).

Step 2. During the phase of authentication stage 1, intercept the Cai (step 5 of Fig. 3) and the Nai (step 7 of Fig. 3) for all $1 \le i \le 20$.

Step 3. For each $1 \le i \le 20$, compute $C'ai=f1(PKax, PKbx, Nai, 0)$ and verify whether $C'ai=Cai$. If so, $rai=0$, else $rai=1$.

Comments.

(1) When the E passively observes a run of the passkey entry protocol, he can collect the PKa, the PKb, the Cai, and the Nai from the public channel and further deduce the ra offline. Therefore, once the host uses the ra again in another new SSP session, the E is able to exploit the MITM attack as Fig. 4. Clearly, if the host chooses another new passkey depended on the ra, the E still possibly derives the new one from the ra.

(2) It needs to point out that Barnickel et al.'s attack [18] is similar to our offline attack. Barnickel et al.'s attack terminates the current SSP session of two devices and reuses the passkey in the next SSP session of two devices. However, our offline attack allows a successful SSP session of the pairing devices and exploits subsequent SSP sessions of them. Hence, our offline attack is not easily detected by the devices owner, compared with Barnickel et al.'s attack.

3.2 Online attack on passkey

In some situations, the host does not change the passkey until the SSP session is successful. The host may believe that the wireless communication errors or the related software flaws cause the failure of the Bluetooth pairing. If the host allows the failures in some degree, the attacker can make use of the online attack to determine the passkey. As shown in Fig. 5, the online attack applies the bit-by-bit strategy to determine the ra. Here, the attacker E interrupts the device A during the run of the passkey entry protocol. And, the E induces both devices, i.e., the A and the B, to start a new SSP session, if the current SSP session fails. To determine the rai for each $1 \le i \le 20$, the online attack detail is described as follows.

When the B generates the Nbi, the E also chooses his Nei (step 3b and step 3a′ in Fig. 5). Then, the B computes the Cbi (step 4b in Fig. 5). At this time, if the rai is determined by the previous run of the protocol, the E computes his $C'ai=f1(PKax, PKbx, Nei, rai)$ (step 4a′ in Fig. 5). Otherwise, the E computes his $C'ai=f1(PKax, PKbx, Nei, 0)$ (also step 4a′ in Fig. 5). Next, the E sends the C′ai and the Nei to the B (step 5′ and step 7′ in Fig. 5). If the B terminates the run of the protocol (step 7a in Fig. 5), then the E knows that the rai is equal to 1 and induces the A to start a new SSP session with the B, else rai=0 when

the rai is an undetermined bit. Moreover, the E continues repeating above rule for the next bit of the ra, if the B does not terminate the current run of the protocol.

To guess the ra, the E intercepts the PKax and the PKbx during the phase of public key exchange. Hence, the E can always compute the C′ai for the unknown rai and further get this rai based on the response of the B. In addition, the E is able to omit the Nbi, accept the correctness of the Cbi, and continuously determine the next unknown bit of the ra, if the B confirms the receiving C′ai from the E.

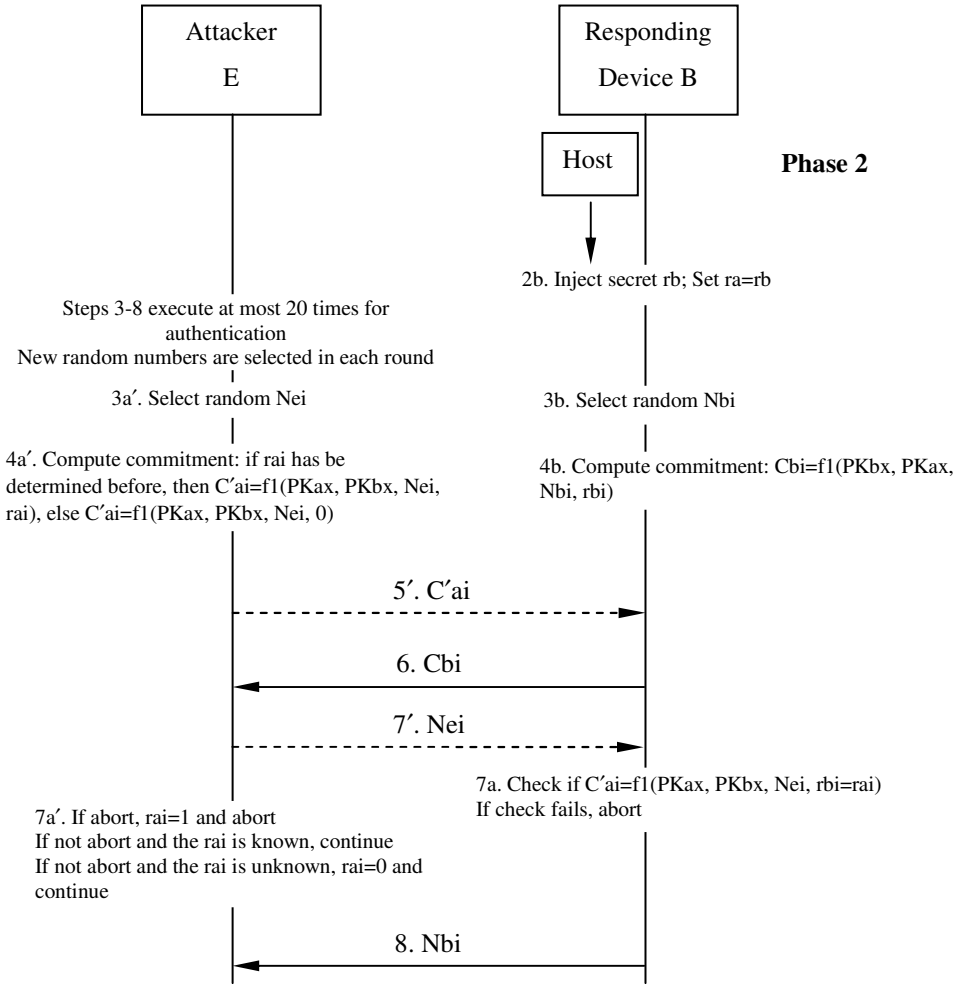

Fig. 5 Online attack on the passkey entry protocol

Let Pr(E) denote the probability that event E occurs. Let n and k be non-negative integers. Assume that the binomial coefficient $\binom{n}{k}$ is the number of different ways of choosing k distinct objects from a set of n distinct objects, where the order of choice is trivial. We further present the property of the online attack.

Theorem 1 Let l be the number of the SSP sessions used in the online attack as Fig. 5. Assume that the ra is a random number with the 20-bit length. Then,

Pr(the ra is determined by the online attack as Fig. 5 before l SSP sessions)

$$= \sum_{k=1}^{l} \binom{20}{k-1} \Big/ 2^{20} + \binom{19}{l-1} \Big/ 2^{20} \quad (1)$$

Proof. In the probability theory, it is well-known that the Bernoulli trial is an experiment with exactly two possible outcomes called success and failure. Obviously, the online passkey attack obeys the Bernoulli trial. That is, if the unknown bit rai is 0, then the outcome of the online passkey attack is treated as success and the attack continues for the next unknown bit of the ra, else the outcome of the attack is regarded as failure and the attacker E need induce the devices to start a new SSP session for the next unknown bit of the ra. Since the ra is treated as a random number with the 20-bit length, Pr(rai=0)=Pr(rai=1)=1/2 for each $1 \le i \le 20$. It means that the failure probability p for guessing any rai is 1/2. According to the fact of the Bernoulli trial, the probability of exactly f failures in the sequence of n=20 such independent trials is

$$\binom{n}{f} \times p^f \times (1-p)^{n-f} = \binom{20}{f} \times (1/2)^f \times (1-1/2)^{20-f} = \binom{20}{f} \Big/ 2^{20} \quad \text{for each } 0 \le f \le 20. \quad (2)$$

We complete the proof and obtain the Eq. (1), since it needs to collect all probabilities of each exactly $0 \le k \le l-1$ failures case and the probability of the failures case when ra20=1 and $k=l$. ∎

Theorem 1 says that Pr(the ra is determined by the offline attack as Fig. 5 before l=13 SSP sessions)= $\sum_{k=1}^{l} \binom{20}{k-1} \Big/ 2^{20} + \binom{19}{l-1} \Big/ 2^{20} = \sum_{k=1}^{13} \binom{20}{k-1} \Big/ 2^{20} + \binom{19}{12} \Big/ 2^{20} \approx 0.9165$.

Therefore, we claim that the online attack can effectively recover the passkey. Note that the practical success probability for guessing the ra is bigger than the theory probability value computed by Eq. (1), because the ra in the Bluetooth standard is a random 6-digit number not a random 20-bit number.

4 Countermeasure on Secure Simple Pairing

4.1 Improved passkey entry protocol

Barnickel et al. [18] gave two methods to remove the reusing passkey attacks in the passkey entry association model. The first method is that the user's passkey should be verified by Bluetooth devices to be at least 20 bits with one as the most significant bit and the devices do not accept the same passkey twice. The second method is that the devices respectively use the DHkey to encrypt the Nai and the Nbi and then instead exchange them during step 7 and step 8 in Fig. 3. We argue that Barnickel et al.'s methods are impractical. The first method requires the devices permanently and securely to record all

previous passkeys of the SSP sessions. The implementation costs of this requirement are quite expensive for the devices. In addition, the users may feel the inconvenience, because the same passkey is not allowed. The second method is still vulnerable to the similar online attack in Section 3.2, if the attacker E shares the Diffie-Hellman key with the device B during the phase of public key exchange. In fact, as described in Fig. 4, the E can exploit the MITM attack to establish the shared $DHkey_2$ with the B. Therefore, to overcome the weaknesses of the reusing passkey, we improve the passkey entry protocol as follows.

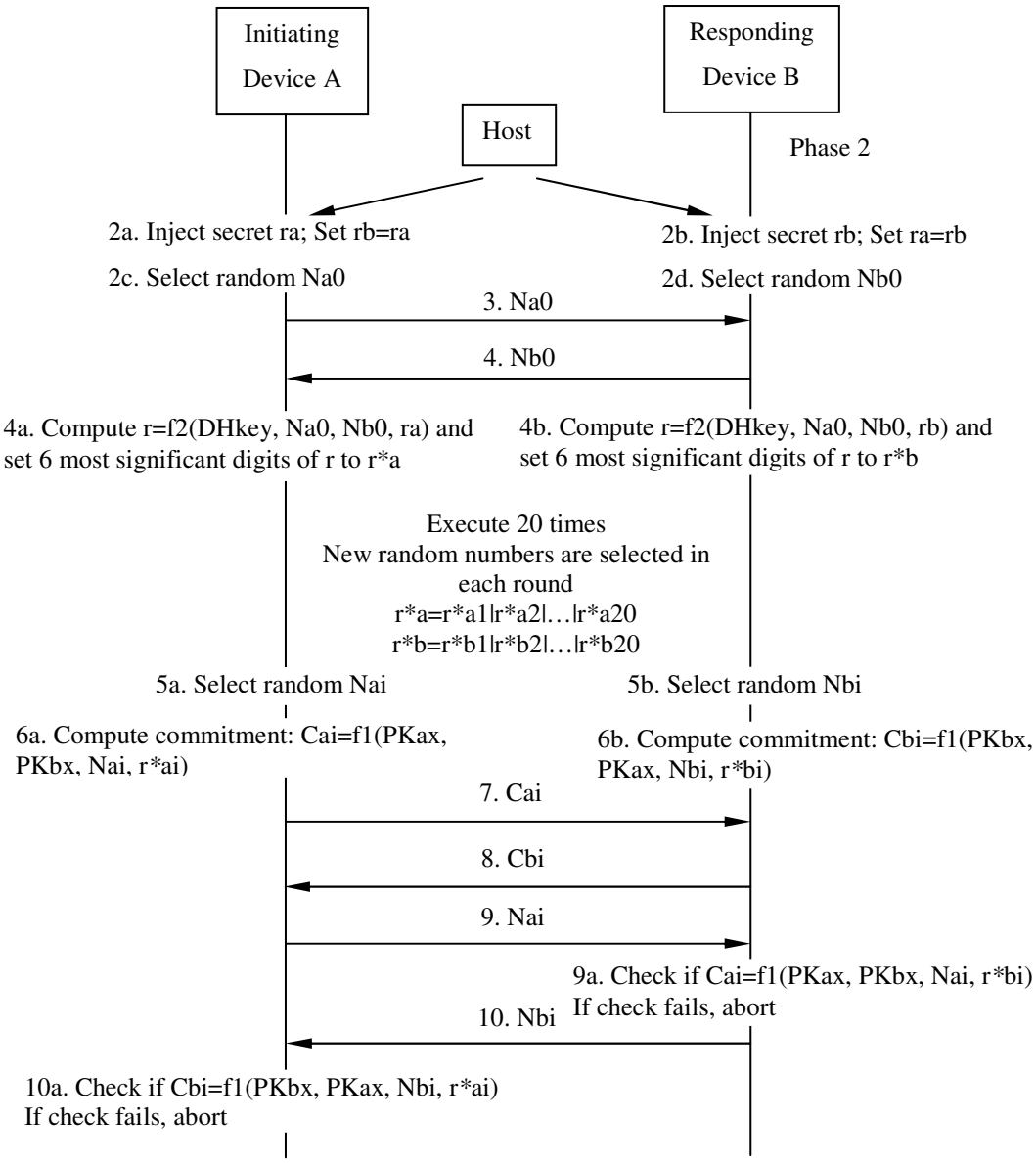

Fig. 6 Improved passkey entry protocol

After the passkey is injected into two devices (step 2a and step 2b in Fig. 6), the device A and the device B respectively generate the random nonce Na0 and the random nonce Nb0 (step 2c and step 2d in Fig. 6). The A and the B then exchange the Na0 and the Nb0

(step 3 and step 4 in Fig. 6). The A computes r=f2(DHkey, Na0, Nb0, ra) and sets the r*a using the 6 most significant digits of the r (step 4a in Fig. 6). Similarly, the B computes r=f2(DHkey, Na0, Nb0, rb) and sets the r*b using the 6 most significant digits of the r (step 4b in Fig. 6). The steps 5 through 10 in improved passkey entry protocol are same as the steps 3 through 8 in original passkey entry protocol (Fig. 3), except that the r*a and the r*b respectively take place of the ra and the rb.

4.2 Security analysis of improved passkey entry protocol

In our attacks on the passkey entry association model, it shows that the attacker takes advantage of the run(s) of the passkey entry protocol to derive host's passkey ra. Then, the attacker is able to launch his MITM attack, once the host injects the ra again. Based on this observation, our countermeasure avoids directly using the ra for the device authentication. Instead, the ra is regarded as a seed of the authentication passkey r during the run of the passkey entry protocol. Moreover, the countermeasure uses the shared secret key DHkey and the cryptographic hash function f2() to guarantee against guessing the ra from the authentication passkey r. At the same time, the countermeasure applies the nonce Na0 from the device A and the nonce Nb0 from the device B to randomly update the authentication passkey r at each run of the passkey entry protocol. In the following, we analyze the security of improved passkey entry protocol in more details.

Case 1 (Concern on offline passkey attack). In fact, the improved passkey entry protocol maintains the same device authentication scheme as that of original passkey entry protocol. That is, the A and the B take turns revealing their commitments Cai and Cbi until the entire ra has been mutually disclosed. Hence, for the attacker E using similar offline passkey attack in Section 3.1, the entire r*a still is available after a complete run of improved passkey entry protocol. But, the r*a is only the 6 most significant digits of the r, where r=f2(DHkey, Na0, Nb0, ra). The E must compromise the ra for launching the MITM attack in the subsequent SSP sessions. It means that the E needs to compute the ra, given the Na0, the Nb0, the r*a, and the f2(). Assume that the f2() is a cryptographic one-way hash function with local one-wayness property [23]. The local one-wayness property means that the E should be difficult to find the remainder input of the f2(), even if part of the input of the f2() is known. For example, if t input bits remain unknown, it should take on average 2^{t-1} cryptographic hash operations to find these bits. We claim that the improved passkey entry protocol prevents the offline attack in Section 3.1, when the f2() maintains the local one-wayness property. That is to say, the E is impossible to reveal the ra from the Na0, the Nb0, the r*a, and the f2(), because the unknown inputs of the f2(), i.e., the DHkey and the ra, are large enough. Obviously, our claim also fits to the case that

the E collects a group of the Na0, the Nb0, and the r*a from several runs of improved passkey entry protocol and attempts to compute the unique ra based on the f2().

Case 2 (Concern on online passkey attack). As described in Fig. 4, the E can establish the $DHkey_2$ shared with the device B, when the phase of public key exchange is finished. Let us consider the E impersonates the A to generate and send his own random Ne0 instead of the Na0 in the run of improved passkey entry protocol. One choice of the E is to simply apply the similar online attack in Section 3.2. Clearly, the E directly guesses the correct 6-digt number r*a with the probability 10^{-6}. If any run of the improved passkey entry protocol fails due to the wrong guess of any bit r*ai for $1 \leq i \leq 20$, the B should generate a new Nb0, compute another $r=f2(DHkey_2, Ne0, Nb0, ra)$, and reset the 6 most significant digits of the new r to the r*a in the next SSP session. Hence, when the E mounts on the similar online attack in Section 3.2, his success probability should be 10^{-6}. This success probability satisfies the requirement of the Bluetooth standard specification [10]. The other choice is that the E controls his Ne0 (step 3 of Fig. 6) and receives the Nb0 from the B (step 4 of Fig. 6), and then further deduces the 6 most significant digits of the r for the device authentication. In this case, the E need predict the 6 most significant digits of the r such that $r=f2(DHkey_2, Ne0, Nb0, ra)$, providing that the ra is unknown and the Nb0 are randomly generated by the B. This success probability also is 10^{-6}, because the f2() has the uniform random distribution property [23]. Note that if the E does not share the $DHkey_2$ with the B, the success probability of launching MITM attack should not be over 10^{-6}. The reason is that the E faces the unknown Diffie-Hellman key but the states of other parameters are unchanged.

As a result, the improved passkey entry protocol overcomes the reusing passkey weaknesses on the original passkey entry protocol. In essence, the secret DHkey added to the ra amplifies the size of the passkey space. Simultaneously, the Na0, the Nb0, and the f2() randomize the r and the r*a for the device authentication. This is the trick behind the improved protocol.

4.3 Performance analysis of improved passkey entry protocol

The improved passkey entry protocol for SSP aims to deploy in different Bluetooth devices and potentially implement in the service level of Bluetooth network systems. Hence, we need carefully evaluate the protocol performance. To be fair, we compare the improved passkey entry protocol with the original passkey entry protocol [10] and Barnickel et al.'s encrypting nonce protocol [18]. The reason is that these protocols are designed for the same security goals of the Bluetooth device and make use of similar basic cryptographic components.

In the implementation complexity concern, the improved protocol requires the random number generator to generate the Nai and the Nbi for each $0 \le i \le 20$, the f2() to derive the authentication passkey r, and the f1() to compute and check the Cai and the Cbi for each $1 \le i \le 20$. Comparatively, the original protocol needs the random number generator to generate the Nai and the Nbi for each $1 \le i \le 20$ and the f1() to compute and check the Cai and the Cbi for each $1 \le i \le 20$. Note that the original protocol also requires the f2() during the phase of link key calculation. It means that the implementation complexity for the improved protocol and the original protocol is same in view of the whole SSP session. However, besides the random number generator and the f1(), Barnickel et al.'s protocol needs extra encryption function to encrypt the Nai and the Nbi for each $1 \le i \le 20$. Therefore, the implementation complexity of Barnickel et al.'s protocol is higher than that of the improved protocol and the original protocol.

As for the communication cost, the improved protocol requires 82 message interactions to finish a run, compared with 80 message interactions in the original protocol. Moreover, the improved protocol need exchange the Na0, the Nb0, the Nai, the Nbi, the Cai, and the Cbi for each $1 \le i \le 20$, while the original protocol requires exchanging the Nai, the Nbi, the Cai, and the Cbi for each $1 \le i \le 20$. Let the Na0 and the Nb0 all be 128 bits. We know that the Nai, the Nbi, the Cai, and the Cbi for each $1 \le i \le 20$ are all 128 bits. Thus, the communication cost of a run is $128 \times 82 = 10496$ bits for the improved protocol and $128 \times 80 = 10240$ bits for the original protocol. The communication cost of a run of Barnickel et al.'s protocol is also 10240 bits, because it merely uses the encrypted nonces to take place of the nonces in the original protocol and those data should have the same bit length.

Consider the computation cost for a run of the protocol. The improved protocol needs 2 cryptographic hash computations of the f2() to derive the authentication passkey r and 80 cryptographic hash computations of the f1() to generate and check the Cai and the Cbi for each $1 \le i \le 20$. We omit the setting operations for the r*a and the r*b in the improved protocol, because the setting operation is trivial, compared with the cryptographic hash computation. By contrast, the original protocol needs 80 cryptographic hash computations of the f1() for processing the Cai and the Cbi for each $1 \le i \le 20$. However, Barnickel et al.'s protocol requires not only 80 cryptographic hash computations of the f1() for the Cai and the Cbi for each $1 \le i \le 20$ but also 40 encryption nonce computations and 40 decryption nonce computations.

For the storage cost, we only consider the long-standing secret values existed in all phases of the SSP session. Clearly, three protocols all require storing the long-standing DHkey. That is, these protocols cost same storage resource.

For comparison purpose, we regard the performance values of the original protocol as the baseline and define the increasing ratio of the communication cost as

$$P_{\text{communication}} = \frac{\text{the communication cost of the target protocol}}{\text{the communication cost of the original protocol}} - 1 \quad (3)$$

and the increasing ratio of the computation cost as

$$P_{\text{computation}} = \frac{\text{the computation cost of the target protocol}}{\text{the computation cost of the original protocol}} - 1. \quad (4)$$

In Table 1, we summarize the performance results of the improved protocol, the original protocol, and Barnickel et al.'s protocol. Here, we assume that the overheads of the encryption or decryption nonce computation are close to those of the cryptographic hash computation. It shows that the total implementation costs of the improved protocol are nearly to those of the original protocol. This is a desirable feature, when the improved protocol fits into the Bluetooth standard. However, the computation cost of Barnickel et al.'s protocol is higher than that of the other two protocols.

Table 1 Performance comparison among the related protocols

Performance index	Our improved protocol	Original protocol	Barnickel et al.'s protocol
Implementation complexity	Medium	Medium	High
Communication cost	10496 bits	10240 bits	10240 bits
$\mathbf{P_{communication}}$	2.5%	0	0
Computation cost	82 H [a]	80 H	80 H+40 E [b]+40 D [c]≈160 H
$\mathbf{P_{computation}}$	2.5%	0	100%
Storage cost	192 or 256 bits	192 or 256 bits	192 or 256 bits

[a.] H denotes the cryptographic hash computation

[b.] E denotes the encryption nonce computation

[c.] D denotes the decryption nonce computation

4.4 Home network application of improved passkey entry protocol

Due to readily available and low-cost feature, an increasing number of Bluetooth devices are connected to the home network according to the Bluetooth standard [9, 10]. In the home network environment, Bluetooth services usually provide the efficient and massive data transmission among the pairing Bluetooth devices. The transmitting data include the text, the picture, the audio, the automation, and the video. The home network should ensure the trusted and uncompromised Bluetooth devices and the corresponding Bluetooth services. The SSP mechanism in the Bluetooth standard is responsible for this

task. In addition, when we build a secure home network system, the SSP mechanism also is the backbone of the Bluetooth solution to cooperate with other kinds of secure schemes [24–29].

Under the home network environments, Bluetooth devices are identifiable and trustable, because they always belong to the home members. Since Phan-Mingard's MITM attack [17] uses the outside device, it is not regarded as a serious threat for SSP under the passkey entry association model. That is to say, the home member would be cautious, when he finds that the unfamiliar Bluetooth device tries to pair his Bluetooth device. By contrast, our proposed MITM attacks are destructive in the case of the home network, because they all run in an undetected manner. The malicious visitor can take his Bluetooth device to intercept the legal SSP session of the passkey entry association model, collect the PKa, the PKb, the Cai, and the Nai, and launch the offline attack on the corresponding passkey (Section 3.1). Alternatively, the malicious visitor's Bluetooth device can hijack the legal SSP session of the passkey entry association model and directly guess the passkey online (Section 3.2). When the home member uses the compromised passkey again, the MITM attack (Fig. 4) should be successful and the malicious visitor may disclose the sensitive information stored in the pairing Bluetooth devices. Unfortunately, the home member does not detect any abnormal state from his pairing Bluetooth devices. The reason is that malicious visitor's Bluetooth device never explicitly takes part in any SSP session.

Note that the home members are perhaps apt to sharing and reusing the passkey. In fact, a constant passkey is convenient to smooth the run of the SSP sessions among Bluetooth devices from the different home members. But, the attacker may exploit this point to launch our MITM attacks. For this reason, the improved passkey entry protocol is designed to prevent the MITM attacks, even if the home members share and reuse their passkey. Therefore, compared with the original passkey entry protocol, the improved passkey entry protocol is more suitable for the home network applications.

5 Conclusions

Bluetooth devices are widely employed in the home network systems. It is important to secure those Bluetooth devices, because they always store and transmit personal sensitive information. In the Bluetooth standard, SSP is an essential security mechanism for Bluetooth devices. Our study dedicates to the MITM attacks on SSP under the passkey entry association model and the corresponding countermeasure. We demonstrated that the MITM attacks are possible, when the host reuses the passkey or the device uses nonrandom passkey generation algorithm in the passkey entry association model. We further improved the passkey entry protocol to prevent the MITM attacks on the passkey

entry association model. The improved protocol can be easily adapted to the Bluetooth standard, because it only employs the basic cryptographic components existed in the Bluetooth standard. Moreover, the improved protocol only increases one round message exchange between the pairing Bluetooth devices and one hash computation for each of them, compared with the original protocol. Hence, the additional implementation cost is insignificant.

Acknowledgments

The work of Dr. Da-Zhi Sun was funded in part by China Scholarship Council and in part by the Open Project of Shanghai Key Laboratory of Trustworthy Computing under Grant No. 07dz22304201402. The authors would like to thank the editor and the reviewers for their valuable suggestions and comments.

References

1. Bluetooth Special Interest Group (SIG), 2017, https://www.bluetooth.org/en-us

2. Hager, C.T., Midkiff, S.F.: An analysis of Bluetooth security vulnerabilities. In: Proceedings of IEEE Wireless Communications and Networking Conference-WCNC 2003, New Orleans, LA, USA, Volume: 3, pp. 1825–1831. IEEE Communications Society (2003)

3. Wong, F.L., Stajano, F., Clulow, J.: Repairing the Bluetooth pairing protocol. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.): Proceedings of the 13th International Security Protocols Workshop-Security Protocols 2005, Cambridge, UK, Lecture Notes in Computer Science, Volume 4631, pp. 31–45. Springer Berlin Heidelberg (2007)

4. Xu, J.F., Zhang, T., Lin, D., Mao, Y., Liu, X.N., Chen, S.W., Shao, S., Tian, B., Yi, S.W.: Pairing and authentication security technologies in low-power Bluetooth. In: Proceedings of IEEE International Conference on Green Computing and Communications-GreenCom, IEEE International Conference on Internet of Things-iThings, IEEE International Conference on Cyber, Physical and Social Computing-CPSCom, Beijing, China, pp. 1081–1085. IEEE Computer Society (2013)

5. Mandal, B.K., Bhattacharyya, D., Kim, T.H.: An architecture design for wireless authentication security in Bluetooth network. International Journal of Security and Its Applications 8(3), 1–8 (2014)

6. Sun, D.Z., Li, X.H.: Vulnerability and enhancement on Bluetooth pairing and link key generation scheme for Security Modes 2 and 3. In: Lam, K.Y., Chi, C.H. (eds.): Proceedings of the 18th International Conference on Information and Communications Security-ICICS 2016, Singapore, Lecture Notes in Computer Science, Volume 9977, pp. 403–417. Springer Berlin Heidelberg (2016)

7. Cope, P., Campbell, J., Hayajneh T.: An investigation of Bluetooth security vulnerabilities. In: Proceedings of the 7th Annual Computing and Communication Workshop and Conference-CCWC, Las Vegas, NV, USA, pp. 1–7. IEEE (2017)

8. Hassan, S.S., Bibon, S.D., Hossain, M.S., Atiquzzaman M.: Security threats in Bluetooth technology. Computers and Security 8(3), in press (2017). DOI: 10.1016/j.cose.2017.03.008
9. Specification of the Bluetooth System, Supplement to the Bluetooth Core Specification, CSSv6, Bluetooth SIG Proprietary, Publication date: Jul. 2015, https://www.bluetooth.com/specifications/adopted-specifications
10. Specification of the Bluetooth System, Covered Core Package Version: 5.0, Master Table of Contents & Compliance Requirements, Bluetooth SIG Proprietary, Publication date: Dec. 2016, https://www.bluetooth.com/specifications/adopted-specifications
11. Padgette, J., Bahr, J., Batra, M., Holtmann, M., Smithbey, R., Chen, L., Scarfone, K.: Guide to Bluetooth security: recommendations of the National Institute of Standards and Technology. National Institute of Standards and Technology (NIST), U.S. Department of Commerce, Special Publication 800-121 Revision 2, May 2017, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf
12. Chang, R., Shmatikov, V.: Formal analysis of authentication in Bluetooth device pairing. In: Proceedings of LICS/ICALP Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis-FCS-ARSPA'07, pp. 45–62. (2007)
13. Suomalainen, J., Valkonen, J., Asokan, N.: Security associations in personal networks: a comparative analysis. In: Proceedings of European Workshop on Security and Privacy in Ad-Hoc and Sensor Networks-ESAS'07, Cambridge, UK, Lecture Notes in Computer Science, Volume 4572, pp. 43–57. Springer Berlin Heidelberg (2009)
14. Lindell, A.Y.: Comparison-based key exchange and the security of the numeric comparison mode in Bluetooth v2.1. In: Fischlin, M. (ed.): Proceedings of the Cryptographers' Track at the RSA Conference-CT-RSA 2009, San Francisco, CA, USA, Lecture Notes in Computer Science, Volume 5473, pp. 66–83. Springer Berlin Heidelberg (2009)
15. Haataja, K., Toivanen, P.: Practical man-in-the-middle attacks against Bluetooth secure simple pairing. In: Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing-WiCOM'08, Dalian, China, pp. 1–5. IEEE (2008)
16. Haataja, K., Toivanen, P.: Two practical man-in-the-middle attacks on Bluetooth secure simple pairing and countermeasures. IEEE Transactions on Wireless Communications 9(1), 384–392 (2010)
17. Phan, R.C.-W., Mingard, P.: Analyzing the secure simple pairing in Bluetooth v4.0. Wireless Personal Communications 64(4), 719–737 (2012)
18. Barnickel, J., Wang, J., Meyer, U.: Implementing an attack on Bluetooth 2.1+ secure simple pairing in passkey entry mode. In: Proceedings of 11th International Conference on Trust, Security and Privacy in Computing and Communications-TrustCom 2012, Liverpool, UK, pp. 17–24. IEEE Computer Society (2012)
19. Albahar, M.A., Keijo, H., Pekka, T.: Bluetooth MITM vulnerabilities: a literature review, novel attack scenarios, novel countermeasures, and lessons learned. International Journal on Information Technologies and Security 8(4), 25–49 (2016)

20. Albahar, M.A., Keijo, H., Pekka, T.: Virtual channel based pairing: a new novel solution structure for Bluetooth pairing. International Journal on Information Technologies and Security 8(4), 51–65 (2016)
21. Albahar, M.A., Keijo, H., Pekka, T.: Towards enhancing just works Bluetooth pairing. International Journal on Information Technologies and Security 8(4), 67–82 (2016)
22. Gajbhiye, S., Sharma, M., Karmkar, S., Sharma, S.: Design, implementation and security analysis of Bluetooth pairing protocol in NS2. In: Proceedings of International Conference on Advances in Computing, Communications and Informatics-ICACCI 2016, Jaipur, India, pp. 1711–1717. IEEE (2016)
23. Menezes, A., Oorschot, P. van, Vanstone, S.: Handbook of Applied Cryptography. CRC Press (1997). Chapter 9
24. Sun, D.Z., Zhong, J.D.: A hash-based RFID security protocol for strong privacy protection. IEEE Transactions on Consumer Electronics 58(4), 1246–1252 (2012)
25. Sun, D.Z., Li, J.X., Feng, Z.Y., Cao, Z.F., Xu, G.Q.: On the security and improvement of a two-factor user authentication scheme in wireless sensor networks. Personal and Ubiquitous Computing 17(5), 895–905 (2013)
26. Yoon, E.J., Kim, C.: Advanced biometric-based user authentication scheme for wireless sensor networks. Sensor Letters 11(9), 1836–1843 (2013)
27. Zhu, H.J., Fang, C.L.H., Liu, Y., Chen, C.L., Li, M.Y., Shen, X.M.S.: You can jam but you cannot hide: defending against jamming attacks for geo-location database driven spectrum sharing. IEEE Journal on Selected Areas in Communications 34(10), 2723–2737 (2016)
28. Li, M.Y., Meng, Y., Liu, J.Y., Zhu, H.J., Liang, X.H., Liu, Y., Ruan, N.: When CSI meets public WiFi: inferring your mobile phone password via WiFi signals. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security-CCS 2016, Vienna, Austria, pp. 1068–1079. ACM (2016)
29. Kim, C., Shin, D., Yang, C.N.: Self-embedding fragile watermarking scheme to restoration of a tampered image using AMBTC. Personal and Ubiquitous Computing, Online First (2017). DOI: 10.1007/s00779-017-1061-x