sensors

MDPI

Article

On Secure Simple Pairing in Bluetooth Standard v5.0-Part II: Privacy Analysis and Enhancement for Low Energy

Da-Zhi Sun [1,*], Li Sun [1] and Ying Yang [2]

1 Tianjin Key Laboratory of Advanced Networking (TANK), Division of Intelligence and Computing, Tianjin University, No. 135, Yaguan Road, Tianjin Haihe Education Park, Tianjin 300350, China
2 Standardization Department, China Aero-Polytechnology Establishment, Aviation Industry of China, No. 7, Jingshun Road, Chaoyang District, Beijing 100028, China
* Correspondence: sundazhi@tju.edu.cn; Tel.: +86-22-2740-1091

Received: 12 May 2019; Accepted: 17 July 2019; Published: 24 July 2019

Abstract: Bluetooth low energy devices are very popular in wireless personal area networks. According to the Bluetooth standard specifications, the low energy secure simple pairing (LESSP) protocol is the process by which the pairing devices negotiate the authenticated secret key. To violate the user privacy, the adversary can perhaps link the runs of the LESSP protocol to the targeted device, which usually relates to the specially appointed user. Hence, we investigate deep into the privacy of the LESSP protocol. Our main contributions are threefold: (1) We demonstrate that the LESSP protocol suffers from privacy vulnerability. That is, an adversary without any secret key is able to identify the targeted device by the LESSP protocol. (2) An improvement is therefore proposed to repair the privacy vulnerability in the LESSP protocol. (3) We develop a formal privacy model to evaluate the privacy vulnerabilities in the LESSP protocol and its improved versions. We further prove that our improvement on the LESSP protocol is private under the privacy model. In addition, the performance evaluation shows that our improvement is as efficient as the LESSP protocol. Our research results are beneficial to the privacy enhancement of Bluetooth systems in wireless personal area networks.

Keywords: Bluetooth standard; low energy; secure simple pairing; cryptographic protocol; untraceability; privacy model

1. Introduction

Due to the development of ubiquitous computing, more and more people carry networking devices, e.g., laptops, smartphones, tablets, and smart watches. These devices are always interconnected via wireless short-range radio frequency communications such as Bluetooth and Wi-Fi. In ubiquitous computing environments, low energy (LE) is an important but challenging requirement for wireless and mobile devices. As the crucial innovation of Bluetooth, the LE form aims to provide low-power and low-cost wireless communications. The LE form was first introduced in Bluetooth standard v4.0 [1] and improved later in Bluetooth standard v4.2 [2]. Bluetooth standard v5.0 [3] further enhances Bluetooth standard 4.2's LE form with up to 4× the range, 2× the speed, and 8× the broadcasting message capacity [4]. Therefore, the LE form in Bluetooth standard v5.0 is treated as an ideal solution of the Internet of Things (IoT) and has been widely deployed in personal or commercial applications, compared with its basic rate/enhanced data rate (BR/EDR) form. The very recent Bluetooth standard v5.1 [5] still maintains the same LE design as Bluetooth standard v5.0.

Bluetooth LE devices have flourished in various person-related fields such as wireless personal area networks. However, these devices may incur severe privacy threats to the users, that is, the adversary can track users' identities through the devices. Consider the scenario where the user connects his smartphone to his nearby laptop via the Bluetooth LE channel. As shown in Figure 1, the adversary can identify the user's identity with his malicious device, because the Bluetooth LE channel is susceptible to radio signal interception. More importantly, the user's exposed identity can be further used as a clue to disclose more sensitive personal information such as medical conditions, financial details, religion preferences, or privileged communications with lawyers. Therefore, a privacy mechanism should be introduced to prevent illegal traceability caused by exploiting the wireless communication processes of the devices.

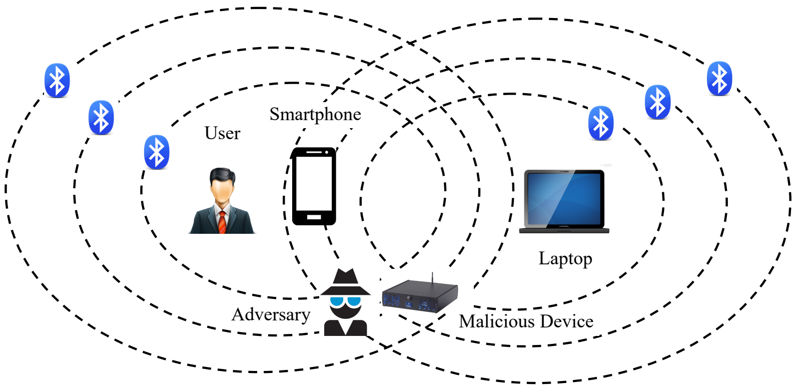


Figure 1. Privacy threat scenario in a typical Bluetooth application.

1.1. Bluetooth LE Security and Privacy

The pairing is the core design to realize the security services in the LE form. In detail, the pairing establishes the secret keys between two Bluetooth devices, and then encrypts a link of the Bluetooth devices and authenticates the Bluetooth devices by using these keys. The pairing method in Bluetooth standards v4.0 and v4.1 [1,6] is referred to as LE legacy pairing. The subsequent Bluetooth standards [2,3,5] adopt LE secure connections pairing as an enhanced version. LE secure connections pairing can be treated as LE secure simple pairing (LESSP). The pairing procedure consists of three phases as follows:

- Phase 1: Pairing feature exchange.
- Phase 2: LESSP protocol or LE legacy pairing protocol.
- Phase 3: Transport specific key distribution (optional).

The pairing feature exchanged in phase 1 determines which of two pairing methods shall be used in phase 2. By default, a LESSP protocol is run in phase 2. A LE legacy pairing protocol is also executable if necessary. Meanwhile, when the LESSP protocol is used, phase 1 needs to choose one of the association models according to the input, output, and display capability. The available association models are the numeric comparison (NC) model, the out of band (OOB) model, the passkey entry (PE) model, and the just works (JW) model. Optionally, phase 3 may be performed to distribute transport specific keys, for example, the identity resolving key (IRK) value.

We can outline the security features of the pairing methods. Since LE legacy pairing has no NC model and does not use the elliptic curve Diffie-Hellman (ECDH) algorithm, it cannot prevent passive eavesdropping except when the OOB model is used. Comparatively, LESSP employs not only the ECDH algorithm to protect from the passive eavesdropping but also the NC, OOB, and PE models to prevent man-in-the-middle (MITM) attacks. Hence, the National Institute of Standards and Technology (NIST) [7] recommends LESSP. However, LESSP should further provide the privacy protection due to the fast development of the personal Bluetooth applications. It is meaningful to ensure that the adversary is unable to identify, track, and associate the targeted devices by exploiting the runs of the pairing protocol. Hence, we will investigate the privacy of LESSP according to Bluetooth standards v5.0 and v5.1 [3,5].

1.2. Related Work

The privacy of the Bluetooth LE form has been widely studied. Gomez et al. [8] described that the LE form offers privacy feature and mitigates the risk that the devices will be tracked by an adversary. Gibbs [9] presented a technical guide to the basis of Bluetooth LE privacy. Wang [10] realized a privacy enhancement protocol over the advertising channels of the LE form. Raza et al. [11] introduced the enhanced privacy feature of the LE form in Bluetooth standard v4.2 and implemented it in the IoT devices. Fawaz et al. [12] investigated the privacy threats from the Bluetooth LE devices and proposed a guardian system to protect the privacy of the users equipped with these devices. AN99209 [13] provided a privacy overview of the link layer of the LE form and compared the privacy features in Bluetooth standards v4.1 and v4.2. Hassan et al. [14] investigated some security and privacy threats in the LE form and the BR/EDR form, such as the pin theft attack, the eavesdropping attack, and the victim device cloning attack. The privacy in the applications of the Bluetooth LE form has also been focused on. Mare et al. [15] proposed a hide-n-sense privacy system, which uses the LE form as a crucial component of the wireless access. Cyr et al. [16] showed that the Fitbit Flex ecosystem provides a reasonable level of privacy for user data, when the adversary observes the LE traffic between the Fitbit devices. Das et al. [17] studied the possible privacy leakage from Bluetooth LE communications between the fitness tracker and the smartphone, i.e., tracking the user location via unchanged Bluetooth LE address and inferring user's current activity through Bluetooth LE traffic analysis. Korolova et al. [18] investigated the possibilities of cross-app tracking using nearby Bluetooth LE devices on Android and iOS. Sivakumaran et al. [19] showed how unauthorized co-located Android applications can access pairing-protected LE data and presented the results of a large-scale static analysis over 18,900 LE Android applications. Cha et al. [20] deployed a privacy framework where the smartphones access the nearby IoT devices via the LE channel. Despite the increasing interest in the investigation of the LESSP security [21–24], previous work did not focus on the LESSP privacy. We examine the LESSP privacy under all possible means of privacy infringement, because the adversary must be expected to exploit any available privacy defects existing in the Bluetooth service.

Moreover, we note that the privacy of the Radio Frequency Identification (RFID) protocols has been well addressed under the formal method. For example, Ha et al. [25] and Juels and Weis [26] proposed formal privacy definitions for the RFID protocols and used them to evaluate the privacy of the RFID protocols. In light of the applicability of RFID-based privacy solutions, Kostakos [27] explored Bluetooth privacy implications. In fact, as the state-of-the-art pairing protocols [28–30], the lack of the formal treatment inspires us to formally analyze the privacy of the LESSP protocol.

1.3. Our Contribution

The LESSP protocol is responsible for the negotiation of the authenticated secret keys between the pairing devices, which usually relate to the specially appointed users. The privacy protection requires that the targeted device's runs of the LESSP protocol be not linked by the adversary. We address this privacy problem on the LESSP protocol. Our main contributions are threefold: (1) We demonstrate that the LESSP protocol suffers from privacy vulnerability, that is, an adversary without any secret key can identify the targeted device, when the targeted device runs the LESSP protocol. (2) An improvement is therefore proposed to repair the existing privacy vulnerability in the LESSP protocol. (3) A formal privacy model is developed to evaluate the privacy of the LESSP protocol and its improved versions. We further prove that our improvement on the LESSP protocol is private under our privacy model. In addition, the performance evaluation shows that our improvement is as efficient as the LESSP protocol.

To the best of our knowledge, previous work [22,28–30] seldom involves the formal model, which can evaluate the security and privacy of the SSP protocol and the LESSP protocol. In Part I [31], we therefore propose a formal security model for the SSP protocol. In this part, a formal privacy model is also presented to define the privacy feature for the LESSP protocol. It needs to be pointed out that the substantial contribution in this part is not only a privacy enhancement on the LESSP protocol but also a privacy model which can be reused to evaluate the privacy of the follow-up LESSP protocols in the future Bluetooth standards. Although the LESSP protocol and the SSP protocol have a slight difference, these two formal models are suitable to evaluate both of them.

1.4. Notation

In Table 1, we list the notations used throughout the rest of the paper. Other notations will be defined where they are first used.

Table 1. Description of notation.

Term	Definition
Π^*	LESSP protocol
Π'	Modified LESSP protocol
Π''	Aggressive LESSP protocol
(Π', Π'')	Our improvement on Π^*
Π	One of Π^*, (Π', Π''), and other LESSP-type protocol
$\Pi^{A,B}$	Bluetooth device A's any Π instance run with Bluetooth device B
X	Bluetooth address of Bluetooth device X
$IOcapX$	IO capabilities of Bluetooth device X
(SKx, PKx)	ECDH private-public key pair of Bluetooth device X
$DHKey$	Diffie-Hellman key
$MacKey$	128-bit key of message authentication code
LTK	128-bit long term key used to generate the contributory session key for an encrypted connection
Nx	Nonce (unique random value) generated by Bluetooth device X
Ex	Check value from Bluetooth device X
rx	Value from Bluetooth device X
$P256()$	ECDH algorithm used to compute $DHKey$
$f5()$	Function used to compute $MacKey$ and LTK
$f6()$	Function used to generate Ex
$Prob(E)$	Probability of event E occurring
$Prob(E' \mid E'')$	Conditional probability of event E' given event E'', which measures probability of event E' occurring, given that event E'' has occurred

2. LESSP Protocol

2.1. Overview

Π^* aims to establish the authenticated LTK between the pairing Bluetooth devices. LTK is subsequently used in the authentication process and the sensitive data transmission process. As shown in Figure 2, Π^* consists of three consecutive phases:

Phase 1: Public key exchange. In this phase, both devices exchange their public keys.

Phase 2: Authentication stage 1. This phase is used to prevent MITM attacks by running one of the three association models, i.e., the NC, OOB, or PE model. This phase also outputs the random nonces for the next phase.

Phase 3: Authentication stage 2. Both devices in this phase confirm that they have generated $MacKey$ and LTK and successfully completed the run of Π^*.

2.2. Detailed Description

Initially, each device generates its own ECDH private-public key pair. That is, the device A and the device B generate (SKa, PKa) and (SKb, PKb) respectively. Bluetooth standard v5.0 [3], p. 2318 states that: "This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so." Π^* is illustrated in Figure 3 and we further describe it as follows.

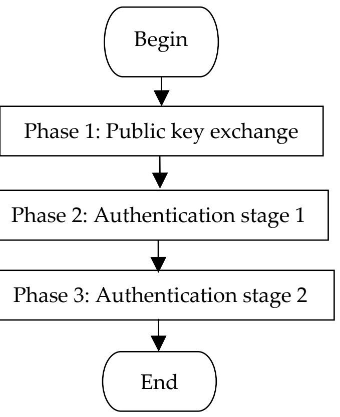

Begin
Phase 1: Public key exchange
Phase 2: Authentication stage 1
Phase 3: Authentication stage 2
End

Figure 2. Flowchart of LESSP protocol.

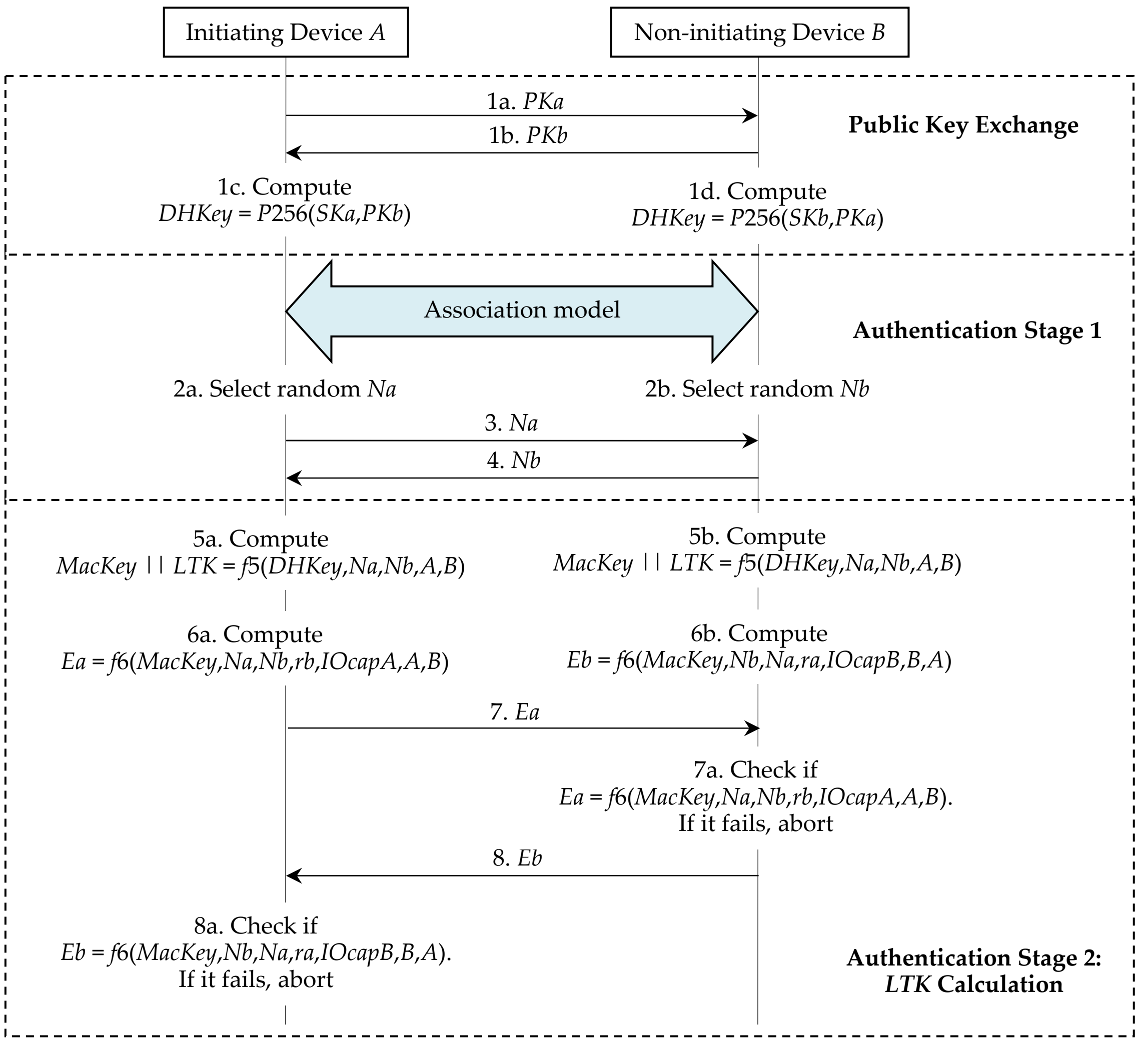

Initiating Device *A*
Non-initiating Device *B*
1a. *PKa*
1b. *PKb*
Public Key Exchange
1c. Compute *DHKey* = P256(*SKa*,*PKb*)
1d. Compute *DHKey* = P256(*SKb*,*PKa*)
Association model
Authentication Stage 1
2a. Select random *Na*
2b. Select random *Nb*
3. *Na*
4. *Nb*
5a. Compute *MacKey* | | *LTK* = *f*5(*DHKey*,*Na*,*Nb*,*A*,*B*)
5b. Compute *MacKey* | | *LTK* = *f*5(*DHKey*,*Na*,*Nb*,*A*,*B*)
6a. Compute *Ea* = *f*6(*MacKey*,*Na*,*Nb*,*rb*,*IOcapA*,*A*,*B*)
6b. Compute *Eb* = *f*6(*MacKey*,*Nb*,*Na*,*ra*,*IOcapB*,*B*,*A*)
7. *Ea*
7a. Check if *Ea* = *f*6(*MacKey*,*Na*,*Nb*,*rb*,*IOcapA*,*A*,*B*). If it fails, abort
8. *Eb*
8a. Check if *Eb* = *f*6(*MacKey*,*Nb*,*Na*,*ra*,*IOcapB*,*B*,*A*). If it fails, abort
Authentication Stage 2: *LTK* Calculation

Figure 3. LESSP protocol Π*.

2.2.1. Phase 1. Public Key Exchange

The initiating device *A* starts a run of Π* and sends its *PKa* to the responding device *B* (step 1a). The responding device *B* replies with its *PKb* (step 1b). The device *A* and the device *B* compute their shared *DHKey* (step 1c and step 1d).

2.2.2. Phase 2. Authentication Stage 1

This phase implements one of the four association models, i.e., the NC, OOB, PE, or JW model. To prevent MITM attacks, the NC, OOB, and PE models require that the user take part in this authentication process. The JW model is the same as the NC model, except that the JW model does not require the user to confirm the commitment checks in both devices. Hence, the JW model cannot prevent MITM attacks. We omit the review of the details of the association models, because the privacy problem in these models will not be discussed. No matter what the association model is used, the device *A* and the device *B* respectively generate the random nonce *Na* and the random nonce *Nb*, and then exchange *Na* and *Nb* for the next phase. Hence, we separate these processes (step 2a, step 2b, step 3, and step 4) from the association models. In addition, Π* requires that both *ra* and *rb* be all set to 0 in the NC and JW models, that both devices generate and exchange the random *ra* and *rb* in the OOB model, and that the user in the PE model enter the same *ra* and *rb* into the devices.

2.2.3. Phase 3. Authentication Stage 2

Each device computes *MacKey* and *LTK* using the shared *DHKey* and the previously exchanged values (step 5a and step 5b). Here, || denotes the bit string concatenation and *A* and *B* are secretly exchanged before. Then the device *A* and the device *B* respectively compute *Ea* and *Eb* by using the newly shared *MacKey* (step 6a and step 6b). Here, *IOCapA* and *IOCapB* are negotiated by the devices during phase 1 of the pairing procedure. The device *A* then transmits its *Ea* to the device *B* (step 7). If the device *B*'s check fails, it indicates that the device *A* has not confirmed the pairing run's authenticity and the pairing run must be aborted by the device *B* (step 7a). The device *B* then transmits its *Eb* to the device *A* (step 8). A failure check also means that the device *B* has not confirmed the pairing run's authenticity and the pairing run should be aborted by the device *A* (step 8a).

3. Privacy Vulnerability on LESSP Protocol

The Bluetooth LE specifications provide some privacy features. For example, the Bluetooth LE solution supports the random resolvable private address (RPA) concept, when the devices need the privacy protection services. The random RPA is generated using IRK and updated after a given interval. And the trusted peer devices resolve the random RPA to an identity address such as the public address or the static address of the device, instead of transmitting the identity address directly. All devices maintain a list of the peer device's identity addresses, the local IRK used to calculate its own random RPA, and the peer device's IRK used to resolve the peer device's random RPA. Due to the random RPA, the adversary is not easy to track the device by its identity address. However, we find that the device can be tracked by exploiting the runs of Π*.

3.1. Privacy Attacks

Each pairing device must transmit its own ECDH public key over the LE channel during every run of Π*. The Bluetooth LE specification allows that the ECDH private-public key pair is generated only once per device and is reused in the multiple runs of Π*. Hence, in order to track the device, the adversary is able to collect these public keys transmitted in each run of Π* and further identify the corresponding runs with a certain device. In detail, the adversary can perform the following steps to track a device *X* via the runs of Π*:

Step 1. The adversary marks the targeted device *X*. In the device *X*'s run of Π*, the adversary eavesdrops on its *PKx* during the public key exchange phase and stores it.

Step 2. In the follow-up runs of Π*, the adversary eavesdrops on each public key during the public key exchange phase. And then, the adversary checks whether the current public key is equal to his local *PKx* stored in step 1. If so, the adversary knows the device *X* takes part in the corresponding run and records the counterpart device.

Figure 4 shows how to create the device *X*'s communication profile by using the above privacy attack. This profile potentially discloses the Bluetooth activities of the device *X* until the change of its *PKx*. A sophisticated adversary can build up a communication constellation for multiple devices by extending the above privacy attack as follows.

Step 1. The adversary marks the multiple targeted devices, and then collects and stores the corresponding public keys in step 1 of the above privacy attack.

Step 2. In step 2 of the above privacy attack, the adversary eavesdrops on the public keys in the subsequent runs of Π* and instead compares them with each of the collected public keys in step 1.

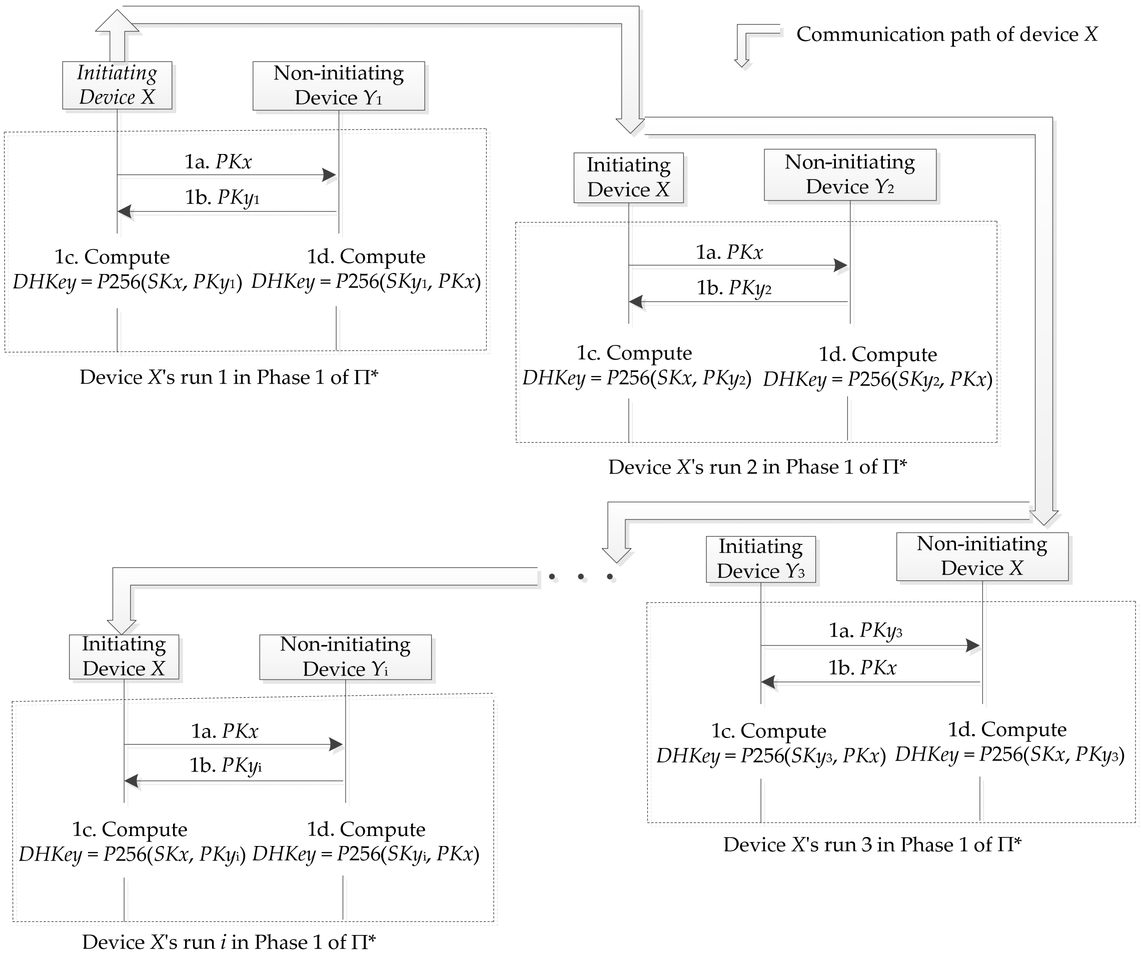

Communication path of device X
Initiating Device X
Non-initiating Device Y1
1a. PKx
1b. PKy1
1c. Compute DHKey = P256(SKx, PKy1)
1d. Compute DHKey = P256(SKy1, PKx)
Device X's run 1 in Phase 1 of Π*
Initiating Device X
Non-initiating Device Y2
1a. PKx
1b. PKy2
1c. Compute DHKey = P256(SKx, PKy2)
1d. Compute DHKey = P256(SKy2, PKx)
Device X's run 2 in Phase 1 of Π*
Initiating Device Y3
Non-initiating Device X
1a. PKy3
1b. PKx
1c. Compute DHKey = P256(SKy3, PKx)
1d. Compute DHKey = P256(SKx, PKy3)
Device X's run 3 in Phase 1 of Π*
Initiating Device X
Non-initiating Device Yi
1a. PKx
1b. PKyi
1c. Compute DHKey = P256(SKx, PKyi)
1d. Compute DHKey = P256(SKyi, PKx)
Device X's run i in Phase 1 of Π*

Figure 4. An example of the tracked device *X*'s communication profile.

Figure 5 gives an example of the extended privacy attack. This extension helps the adversary to infer the user's behavior characteristics and the multiple users relationship via a great deal of the tracked devices.

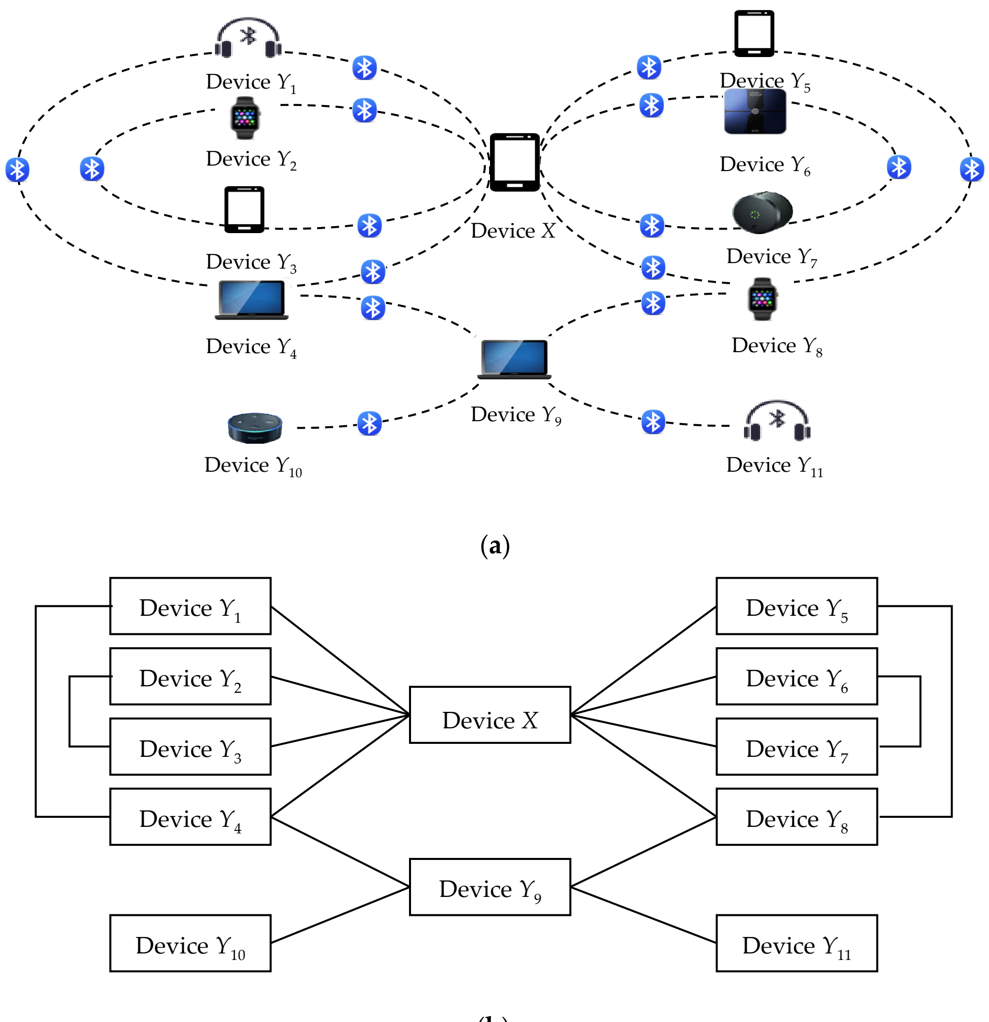


Figure 5. An example of the extended privacy attack: (**a**) The communication activities of multiple tracked devices; (**b**) The tracked devices' communication constellation derived from Figure 5a.

3.2. Technical Remarks and Experimental Confirmation

The proposed privacy attacks are practical due to the following technical factors:

1. To defend itself against the proposed attacks, the device may update its public key in each run of Π^*. We argue that the device tends to reuse the public key during the multiple runs of Π^*. The device needs to expend more system resources, especially more device energy, if the ECDH private-public key pair is frequently updated. However, the device energy cost is just the critical factor in the LE form.
2. The proposed attacks merely exploit the vulnerability in the public key exchange phase of Π^*. We know that the public key exchange phase is the same no matter what the NC, OOB, PE, or JW model is selected. Hence, the proposed attacks are regarded as the broad-spectrum tracking ways.
3. The proposed attacks can be performed in the off-the-shelf Bluetooth tools without the need for any hardware retrofit, because they only eavesdrop on the transmitted messages over the LE channel.

Moreover, as shown in Figure 6, we build an experiment platform to verify the proposed attacks. We use a commercially available sniffer Ubertooth One USB dongle under the Ubuntu 14.04.1 in the Oracle VM VirtualBox and test the LESSP protocol implemented by the LightBlue apps of iOS devices.

Figure 7 shows the pairing public key packets sniffed by Ubertooth One USB dongle during two runs of the LESSP protocol. Here, we can track the device *X*.

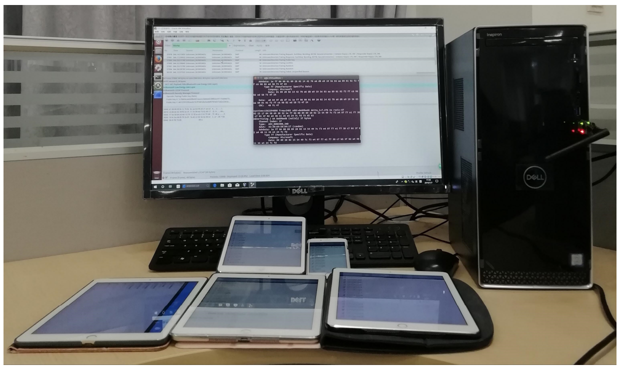
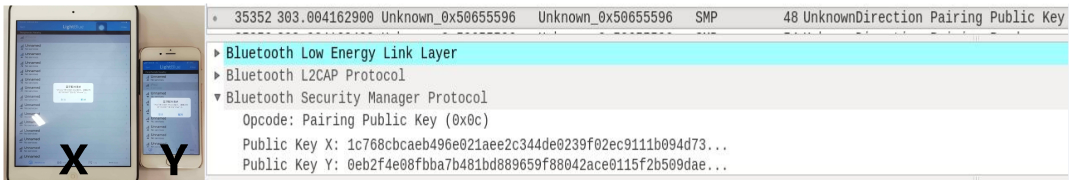
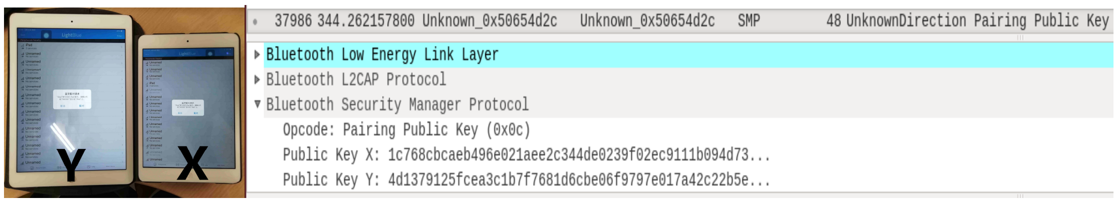


Figure 6. Privacy experiment platform for LE Bluetooth devices.



```
•  35352 303.004162900 Unknown_0x50655596   Unknown_0x50655596   SMP        48 UnknownDirection Pairing Public Key
▸ Bluetooth Low Energy Link Layer
▸ Bluetooth L2CAP Protocol
▾ Bluetooth Security Manager Protocol
    Opcode: Pairing Public Key (0x0c)
    Public Key X: 1c768cbcaeb496e021aee2c344de0239f02ec9111b094d73...
    Public Key Y: 0eb2f4e08fbba7b481bd889659f88042ace0115f2b509dae...
```

(**a**)



```
•  37986 344.262157800 Unknown_0x50654d2c   Unknown_0x50654d2c   SMP        48 UnknownDirection Pairing Public Key
▸ Bluetooth Low Energy Link Layer
▸ Bluetooth L2CAP Protocol
▾ Bluetooth Security Manager Protocol
    Opcode: Pairing Public Key (0x0c)
    Public Key X: 1c768cbcaeb496e021aee2c344de0239f02ec9111b094d73...
    Public Key Y: 4d1379125fcea3c1b7f7681d6cbe06f9797e017a42c22b5e...
```

(**b**)

Figure 7. (**a**) Test 1 and its corresponding pairing public key packet (public key *X*: 1c768cbcaeb496e 021aee2c344de0239f02ec9111b094d7302d017175e50b4a5, public key *Y*: 0eb2f4e08fbba7b481bd8 89659f 88042ace0115f2b509dae3c8c15510a37d5ba); (**b**) Test 2 and its corresponding pairing public key packet (public key *X*: 1c768cbcaeb496e021aee2c344de0239f02ec9111b094d7302d017175e50b4a5, public key *Y*: 4d1379125fcea3c1b7f7681d6cbe06f9797e017a42c22b5e4a30ce5841fb73d6).

4. Improvement on LESSP Protocol

4.1. Design Frame

Informally, the adversary fails to track the device *X* by implementing the proposed attacks, if the device *X* randomly generates a new private-public key pair (*SKx*, *PKx*) for every run of Π^*. However, this update operation is inefficient. To securely reuse the key pair, our idea is to create a pairing information table, that is, each device *X* secretly maintains a table for its previous pairing runs. As shown in Figure 8, we require that each device *X*'s table store its previous pairing information with the devices Y_1, Y_2, … , and Y_m. Each record in the table needs to contain the device address and the corresponding *DHKey*. Once any a record compromises or expires, the device *X* can remove it from its table. Certainly, the device *X* should first choose the information of the frequently paired devices into the table.

Y_1's device address	Y_1's corresponding *DHKey*
Y_2's device address	Y_2's corresponding *DHKey*
$\vdots$	$\vdots$
Y_m's device address	Y_m's corresponding *DHKey*

Figure 8. The device *X*'s pairing information table.

When any two devices *X* and *Y* begin to pair, they first search their pairing information tables indexed the device address. If both devices find the other one in their pairing information tables, then they implement an aggressive LESSP protocol Π″; else they implement a modified LESSP protocol Π′, where Π′ and Π″ are respectively proposed in the following two subsections. Figure 9 describes the process flow of (Π′, Π″). Here, similar to Π*, both Π′ and Π″ should implement the association model to verify the public keys. However, for simplicity, we omit it, because the privacy of the association model is not considered.

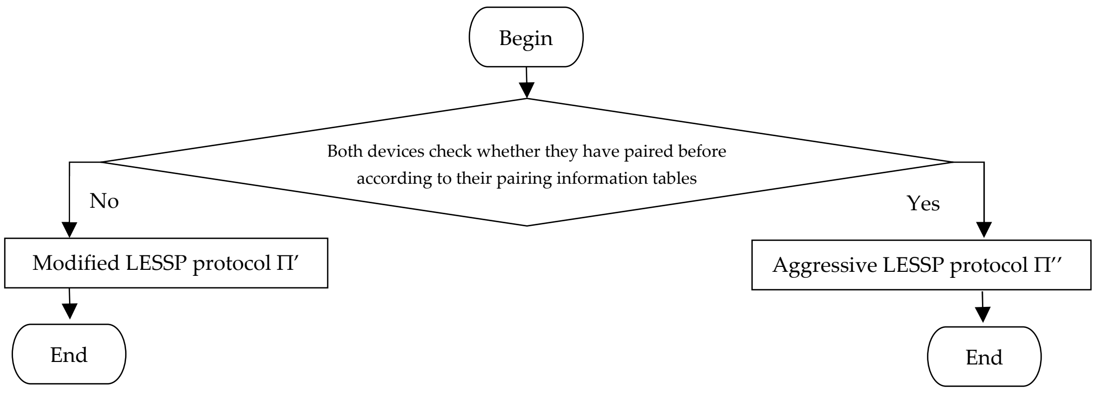

Begin
Both devices check whether they have paired before according to their pairing information tables
No
Yes
Modified LESSP protocol Π′
Aggressive LESSP protocol Π″
End
End

Figure 9. Flowchart of our improvement (Π′, Π″).

4.2. Modified LESSP Protocol

As shown in Figure 10, Π′ is almost the same as Π* in Figure 3. The different points of two protocols are as follows:

(1) Π′ mandatorily requires that the device *A* and the device *B* randomly generate the new private-public key pairs.
(2) In the table update phase of Π′, the device *A* and the device *B* may store new data into their pairing information tables.

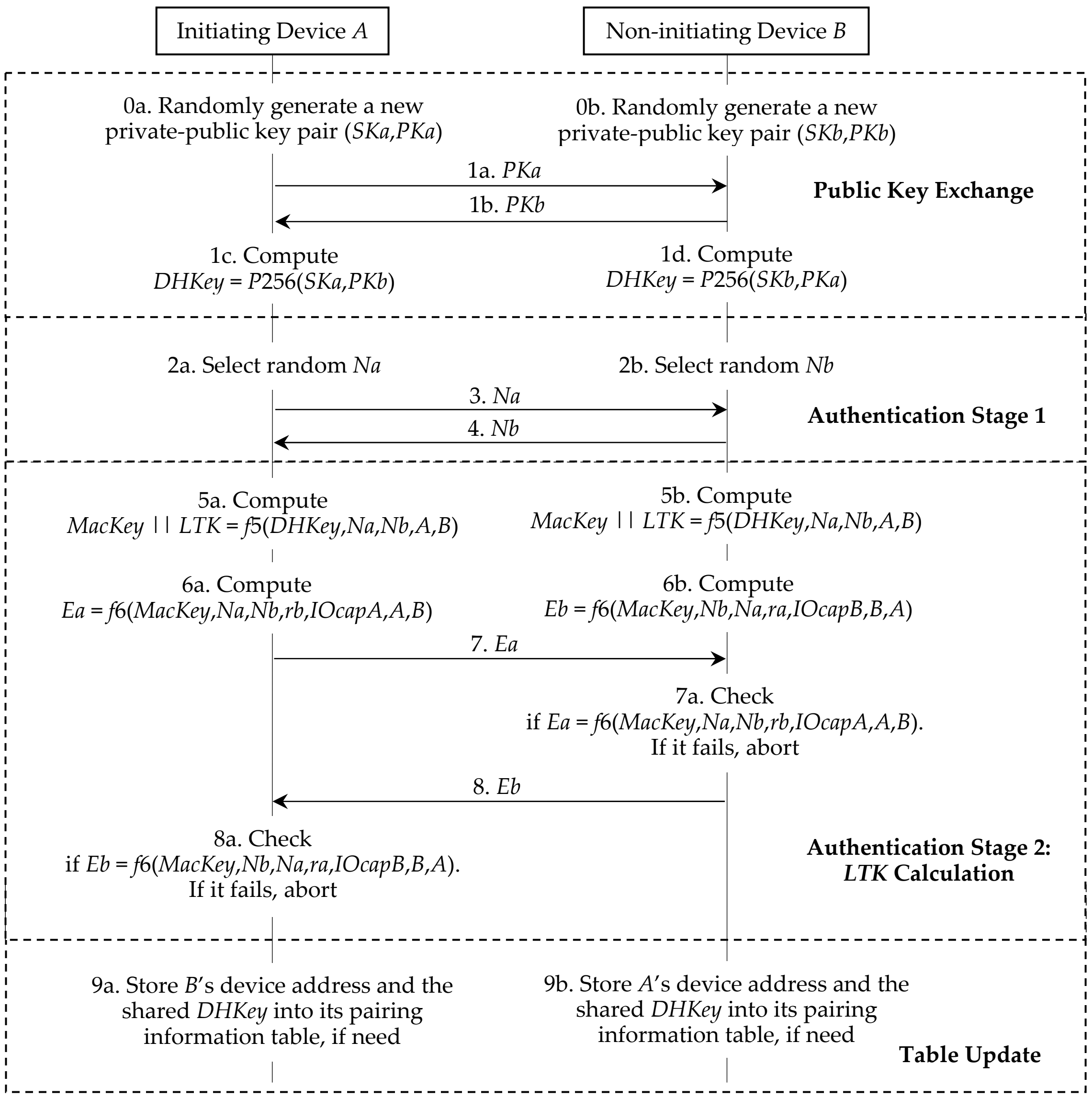

Initiating Device *A*
Non-initiating Device *B*
0a. Randomly generate a new private-public key pair (*SKa*,*PKa*)
0b. Randomly generate a new private-public key pair (*SKb*,*PKb*)
1a. *PKa*
1b. *PKb*
Public Key Exchange
1c. Compute *DHKey* = P256(*SKa*,*PKb*)
1d. Compute *DHKey* = P256(*SKb*,*PKa*)
2a. Select random *Na*
2b. Select random *Nb*
3. *Na*
4. *Nb*
Authentication Stage 1
5a. Compute *MacKey* || *LTK* = *f*5(*DHKey*,*Na*,*Nb*,*A*,*B*)
5b. Compute *MacKey* || *LTK* = *f*5(*DHKey*,*Na*,*Nb*,*A*,*B*)
6a. Compute *Ea* = *f*6(*MacKey*,*Na*,*Nb*,*rb*,*IOcapA*,*A*,*B*)
6b. Compute *Eb* = *f*6(*MacKey*,*Nb*,*Na*,*ra*,*IOcapB*,*B*,*A*)
7. *Ea*
7a. Check if *Ea* = *f*6(*MacKey*,*Na*,*Nb*,*rb*,*IOcapA*,*A*,*B*). If it fails, abort
8. *Eb*
8a. Check if *Eb* = *f*6(*MacKey*,*Nb*,*Na*,*ra*,*IOcapB*,*B*,*A*). If it fails, abort
Authentication Stage 2: *LTK* Calculation
9a. Store *B*'s device address and the shared *DHKey* into its pairing information table, if need
9b. Store *A*'s device address and the shared *DHKey* into its pairing information table, if need
Table Update

Figure 10. Modified LESSP protocol Π′.

4.3. Aggressive LESSP Protocol

As shown in Figure 11, Π″ can be treated as a simplified version of Π* in Figure 3. In Π″, although the device *A* and the device *B* reuse the shared *DHKey* in their pairing information tables, they still need to randomly generate *PKa* and *PKb* and then exchange them during the public key exchange phase. Note that *PKa* and *PKb* are directly selected from the defined elliptic curve [3,5] but are not computed from *SKa* and *SKb* as in Π* and Π′. The authentication stage in Π″ is similar to the authentication stage 2 in Π*. The only difference from the authentication stage is that the device *A* and the device *B* generate *Na* and *Nb* respectively and then exchange them (step 2a, step 2b, step 3, and step 4), that is, these steps appeared in the authentication stage 1 in Π*.

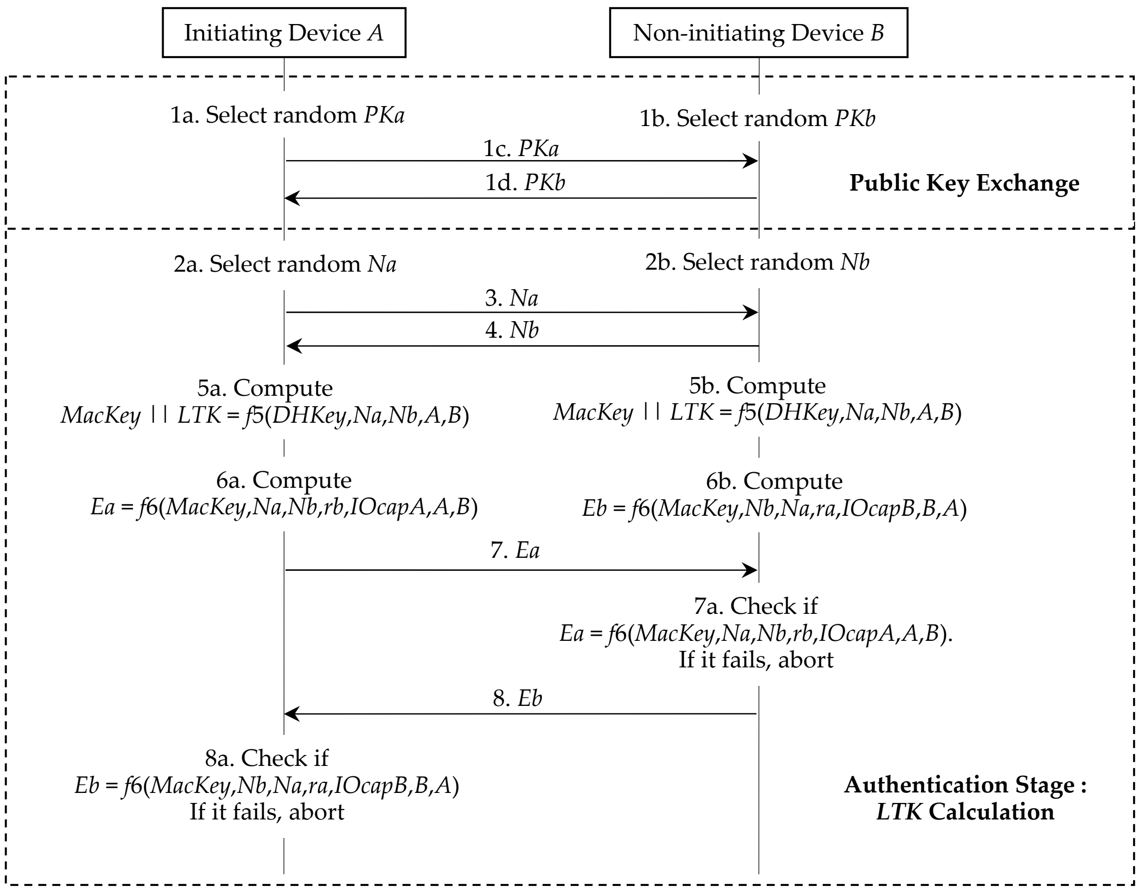

Initiating Device *A*
Non-initiating Device *B*
1a. Select random *PKa*
1b. Select random *PKb*
1c. *PKa*
1d. *PKb*
Public Key Exchange
2a. Select random *Na*
2b. Select random *Nb*
3. *Na*
4. *Nb*
5a. Compute *MacKey* | | *LTK* = *f*5(*DHKey*,*Na*,*Nb*,*A*,*B*)
5b. Compute *MacKey* | | *LTK* = *f*5(*DHKey*,*Na*,*Nb*,*A*,*B*)
6a. Compute *Ea* = *f*6(*MacKey*,*Na*,*Nb*,*rb*,*IOcapA*,*A*,*B*)
6b. Compute *Eb* = *f*6(*MacKey*,*Nb*,*Na*,*ra*,*IOcapB*,*B*,*A*)
7. *Ea*
7a. Check if *Ea* = *f*6(*MacKey*,*Na*,*Nb*,*rb*,*IOcapA*,*A*,*B*). If it fails, abort
8. *Eb*
8a. Check if *Eb* = *f*6(*MacKey*,*Nb*,*Na*,*ra*,*IOcapB*,*B*,*A*) If it fails, abort
Authentication Stage : ***LTK* Calculation**

Figure 11. Aggressive LESSP protocol Π″.

5. Privacy Analysis

5.1. Threats

Π′s messages are transmitted over the wireless LE channel and therefore are easily tapped by the adversary. Moreover, the possibility of the message loss over the LE channel is higher than that over the wired channel due to mobile system malfunction or wireless communication errors. Hence, we assume that the devices and their LE channel can be controlled by the adversary. In detail, we classify two types of adversaries as follows:

- *Passive adversary (PA)*. A PA eavesdrops on all messages generated by the run of Π. The PA tries to find out *LTK* or other useful information of the targeted devices. However, the PA cannot tamper with any message during the run of Π.
- *Active adversary (AA)*. Besides acting as a PA, an AA can additionally inject, modify, and block the messages transmitted by the devices. When the devices run Π, the AA may impersonate a device by sending his fabricated messages.

According to the security and privacy requirements [1–3,5,6], we summarize that Π should be particularly able to defeat against the following attacks:

1. Eavesdropping attack. During the run of Π, the PA or AA eavesdrops on the devices' messages over the LE channel and tries to derive LTK. When LTK is revealed, the PA can decrypt the encrypted data from the devices and the AA may perform other enhanced attacks, such as the replay attack and the spoofing attack.
2. MITM attack. The MITM attack occurs, when a user wants to connect two devices but instead of connecting directly with each other they unknowingly connect to the AA's device that plays

the role of the device they are attempting to pair with. To achieve the MITM attack, the AA may replay the message(s) in the previous run of Π or forge the message(s) for the current run of Π.

3. Message loss. When the AA blocks the LE service by jamming, the messages transmitted by the devices can be interrupted. During the run of Π, the message loss may cause the data de-synchronization in the pairing information tables. We call it as the de-synchronization attack.
4. Tracking. The PA or AA may track the devices by observing their protocol interactions. For perfect untraceability, Π must ensure that the messages emitted by the targeted device should not be distinguishable from the messages emitted by other devices.

The LESSP protocol makes use of the association models and the ECDH algorithm to prevent the eavesdropping attack and the MITM attack. In Part I [31], we show that the SSP protocols can withstand the eavesdropping attack and the MITM attack under the formal security model. In fact, with the similar security model, (Π', Π'') described in Section 4 can be proved secure against the eavesdropping attack and the MITM attack. For simplicity, we do not discuss this security feature of (Π', Π''), because it can be similarly analyzed but nevertheless with a lot of background knowledge of our security model [31]. Next, we propose a formal privacy model to address the de-synchronization attack and the tracking attack.

5.2. Privacy Model

Let k or 1^k be the security parameter. A real-valued function f: $N \to [0, 1]$ is negligible if for every polynomial $pl()$ there exists an integer N such that for every $k > N$, $f(k) < 1/pl(k)$ holds. In the following, we denote an arbitrary negligible function by *negl*. For the sake of readability, we concentrate on asymptotic complexities and privacy, although all our results can be written with more precise bounds.

The Bluetooth pairing system includes a polynomial number of the devices named as 1, 2, . . . , and n. Each device is a probabilistic interactive Turing machine with an independent source of randomness and unlimited internal storage. Any different devices $A, B \in \{1, 2, \dots, n\}$ are allowed to run a Π multiple times with the polynomial bound. The goal of the PA and AA is to identify the targeted device by exploiting the runs of Π among the devices. Our formal privacy model generally simulates the pairing process of the devices controlled by the adversary and excludes the potential attacks using the privacy definition. Our model can be slightly adjusted to fit any particular deployment model of the Bluetooth pairing system.

5.2.1. Modeling Adversarial Power

To compromise the privacy of Π, the adversary is able to access the Π's instances run by the devices. This adversarial behavior is formalized by allowing the adversary to interact freely with a group of oracles. The oracles provided for the adversary are defined as follows:

1. Init (A, B): This oracle call initializes the fresh partnered instances $\Pi^{A,B}$ and $\Pi^{B,A}$, where $1 \le A \le n$, $1 \le B \le n$, and $A \ne B$. For convenience, we assume that A is always the initiating device and B is always the non-initiating device. For (Π', Π''), the Init oracle obeys the following rule. If either the device A or the device B cannot find the counterpart's device address in its pairing information table, two fresh instances $\Pi'^{A,B}$ and $\Pi'^{B,A}$ are initialized; else two fresh instances $\Pi''^{A,B}$ and $\Pi''^{B,A}$ are initialized.
2. Send ($\Pi^{A,B}$, M) or Send ($\Pi^{B,A}$, M): When this Send oracle is called, it sends the message M to A's $\Pi^{A,B}$ or B's $\Pi^{B,A}$. The output of the Send oracle is whatever message $\Pi^{A,B}$ or $\Pi^{B,A}$ would send after receiving the message M according to the current run progress of $\Pi^{A,B}$ or $\Pi^{B,A}$. Especially, the Send oracle outputs the first message of the current run of Π, if Send ($\Pi^{A,B}$, null) is invoked and $\Pi^{A,B}$ is a new instance. The Send oracles simulate the AA by injecting the messages into the runs of Π or modifying the messages transmitted by the devices during the runs of Π. Moreover, the device should terminate the current run of Π, if the message M into the Send oracle is invalid.

Hence, the Send oracle can also simulate the AA by blocking the runs of Π by sending the error messages.

3. Execute (A, B): When this Execute oracle is called, a complete run between the partnered instances $\Pi^{A,B}$ and $\Pi^{B,A}$ is performed. The output of the Execute oracle is a transcript of Π, i.e., the complete series of messages exchanged by the partnered instances $\Pi^{A,B}$ and $\Pi^{B,A}$. The Execute oracle simulates the PA's ability to passively eavesdrop on the protocol runs. To protect the user privacy, the adversary should learn nothing of the device from such oracle calls.
4. Result $\Pi^{A,B}$ or Result $(\Pi^{B,A})$: If $\Pi^{A,B}$ or $\Pi^{B,A}$ completes successfully, the output of the Result oracle returns 1; else the output of the Result oracle results 0. The adversary may have access to the partial protocol output, e.g., the observation whether a Bluetooth-activated door opens or not. The Result oracle therefore simulates the adversary knowing whether the device succeeds to be paired with the counterpart device.
5. Reveal $\Pi^{A,B}$ or Reveal $(\Pi^{B,A})$: This call outputs the link keys, i.e., *MacKey* and *LTK*, in the device A's $\Pi^{A,B}$ or the device B's $\Pi^{B,A}$. If *MacKey* and *LTK* do not exist, the Reveal oracle outputs a null. This Reveal oracle simulates the improper exposure of the link keys. To defeat tracking, the protocol needs to ensure independence of the link keys in different instances.
6. Corrupt $(\Pi^{A,B})$ or Corrupt $(\Pi^{B,A})$: This call outputs *DHKey* in $\Pi^{A,B}$ or $\Pi^{B,A}$, if it exists. Otherwise, the Corrupt oracle outputs a null. This Corrupt oracle simulates the improper exposure of the Diffie-Hellman key.
7. Test (A, B): This call is used to define the privacy of Π and does not simulate any real adversarial ability. The adversary is only allowed to query the Test oracle once. To answer the query, the Test oracle flips a fair random coin $b \in \{0, 1\}$. If $b = 0$, the Test oracle calls Execute (A, B) and returns the output of Execute (A, B) to the adversary; else the Test oracle itself generates a random transcript of Π and returns it to the adversary. Here, we demand that the random transcript of Π has the same bit length with that of the output of Execute (A, B). The adversary is required to guess the random bit b according to the output of Test (A, B). For (Π', Π''), the Test oracle is specifically implemented as follows.

When $b = 0$, the Test oracle calls Execute (A, B). To be specific, the device A and the device B respectively search their own pairing information tables. If the other device addresses exist in their tables, then Π'' is run and the complete series of the corresponding transmitted messages is output. Otherwise, the complete series of the corresponding transmitted messages is output through the run of Π'.

When $b = 1$, the Test oracle generates *SKa*, *PKa*, *SKb*, *PKb*, *Na*, *Nb*, *Ea*, and *Eb* according to the specification of Π' and returns *PKa*, *PKb*, *Na*, *Nb*, *Ea*, and *Eb* to the adversary.

5.2.2. Defining Privacy

In our privacy model, violating the privacy of Π informally means that the adversary distinguishes a transcript of Π generated by a device from a random transcript without any relation to the device. It is reminiscent of the polynomially indistinguishable chosen-plaintext attack [32] in a cryptosystem security game.

Let $\mathcal{A}$ denote the adversary. As shown in Table 2, we present a privacy game $\mathcal{PG}_{\mathcal{A},\Pi}[n, k]$, where n is the number of the devices in the Bluetooth pairing system and k is the security parameter of Π. The Bluetooth pairing system is set up in the system setup phase. Each device in the Bluetooth pairing system uses Π to pair with each other. During the learning phase, $\mathcal{A}$ is allowed to invoke the Init, Send, Execute, Result, Reveal, and Corrupt oracles defined in Section 5.2.1. Here, assume that $\mathcal{A}$ has the polynomial limits of its computational power and the polynomial bounds of the number of oracles calls. In the challenge phase, $\mathcal{A}$ arbitrarily chooses two devices, i.e., A and B, without the compromise of their shared *DHKey*, and then calls Test (A, B). The task of $\mathcal{A}$ is to guess the random bit b according to the output of Test (A, B). $\mathcal{A}$ should determine b by distinguishing the transcript of Π, i.e., the output of Execute (A, B), from a random transcript generated by the Test oracle. Let $\mathcal{A}[n, k]$ denote $\mathcal{A}$ with n and k. Π is defined to be private if no $\mathcal{A}[n, k]$ has a non-negligible advantage in successfully guessing the random bit b in the Test oracle. Now, the privacy of Π can be defined as follows.

Table 2. Bluetooth privacy game for the LESSP-type protocols.

Untraceability on Bluetooth Device
Bluetooth Privacy Game $\mathcal{PG}_{\mathcal{A},\Pi}[n, k]$
Phase 1. System setup
1. Select a polynomial number of the Bluetooth devices named as 1, 2, … , and n. Determine the bit length of *DHKey* denoted as k_1 and the bit length of *MacKey* denoted as k_2. Let $k = \min(k_1, k_2)$ be the security parameter.
2. (Optional) Set the available storage capacity for the pairing information table in the Bluetooth device.
Phase 2. Learning
3. For any Bluetooth device in {1, 2, … , n}, the adversary $\mathcal{A}$ can call the oracles defined in Section 5.2.1. The following oracles are polynomially invoked with interleaved order.
3.1. $\mathcal{A}$ calls the oracles O_{PA} = {Execute, Result} to act as the PA.
3.2. $\mathcal{A}$ calls the oracles O_{AA} = {Init, Send, Reveal, Corrupt} to act as the AA.
Phase 3. Challenge
4. $\mathcal{A}$ selects a challenge Bluetooth device $A \in \{1, 2, \dots, n\}$ and any other Bluetooth device $B \in \{1, 2, \dots, n\}$. Here, the oracle Corrupt ($\Pi^{A,B}$) or the oracle Corrupt ($\Pi^{B,A}$) is not allowed during the learning phase, if the compromised *DHKey* exists in both devices' pairing information tables.
5. $\mathcal{A}$ calls the oracle Test (A, B).
6. $\mathcal{A}$ outputs his guessing bit b'.
$\mathcal{A}$ wins if $b = b'$.

Definition 1 (k-Privacy). Π *is k-privacy if for any probabilistic polynomial time* $\mathcal{A}[n, k]$ *there exists a negligible function negl such that:*

$$Prob(\mathcal{PG}_{\mathcal{A},\Pi}[n,k] succeeds\ in\ guessing\ b) < \frac{1}{2} + negl(k). \quad (1)$$

We show that the LESSP protocol fails to pass the privacy evaluation under the above privacy game. Consider $\mathcal{PG}_{\mathcal{A},\Pi^*}[n, k]$. During the learning phase, the adversary $\mathcal{A}$ is able to select any two different devices $A, B \in \{1, 2, \dots, n\}$, call the oracle Execute (A, B), and obtain its output, i.e., *PKa*, *PKb*, *Na*, *Nb*, *Ea*, and *Eb*. During the challenge phase, $\mathcal{A}$ continues to select two uncorrupted devices A and B, and then calls the oracle Test (A, B). The oracle Test (A, B) returns *PKa*, *PKb*, *Na*, *Nb*, *Ea*, and *Eb* to $\mathcal{A}$. If the previous *PKa* in the learning phase is equal to the current *PKa*, $\mathcal{A}$ determines $b' = 0$ in step 6 of $\mathcal{PG}_{\mathcal{A},\Pi^*}[n, k]$; else $\mathcal{A}$ guesses $b' = 1$ in step 6 of $\mathcal{PG}_{\mathcal{A},\Pi^*}[n, k]$. Assume that p^* is the probability of reusing (*SKa*, *PKa*) in the device A's next run of Π^*. We have:

$$\begin{gathered} Prob(\mathcal{PG}_{\mathcal{A},\Pi^*}[n,k] \text{ succeeds in guessing } b) = \\ Prob(\mathcal{PG}_{\mathcal{A},\Pi^*}[n,k] \text{ succeeds in guessing } b \mid b = 0)\ Prob(b = 0) + \\ Prob(\mathcal{PG}_{\mathcal{A},\Pi^*}[n,k] \text{ succeeds in guessing } b \mid b = 1)\ Prob(b = 1) \approx p^* \tfrac{1}{2} + \tfrac{1}{2} = \tfrac{1}{2} + \tfrac{p^*}{2}. \end{gathered} \quad (2)$$

Because p^* is not negligible, we conclude that Π^* violates the privacy requirement in Definition 1.

5.3. Privacy Result of Our Improvement and Its Proof

The adversary may initiate the de-synchronization attack on the device and further observe the abnormal run of Π to track this device. In (Π', Π''), the only way for the attack to be successful is to de-synchronize the data of the pairing information tables between the pairing devices. However, in Figure 9, we see that the devices should implement Π', if a device fails to share *DHKey* in the previous run. Clearly, (Π', Π'') can prevent the adversary from guessing the random bit b by the de-synchronization attack, because it does not make the device enter the abnormal run. Hence, we do not consider the de-synchronization attack in the following discussion. We need two well-known cryptographic assumptions to prove the privacy result of (Π', Π''). For the reader's convenience, we rewrite them as follows:

Definition 2 (Definition 3 in [31]). *Let gen(1^k) be a parameter generation algorithm that outputs the description of a group G, its generator $P \in G$, and its order q. We say the Decisional Diffie-Hellman (DDH) problem is hard relative to G if for all probabilistic polynomial time algorithms D there exists a negligible function negl such that:*

$$|Prob(D\ (gen\ (1^k), a{\cdot}P, b{\cdot}P, ab{\cdot}P) = 1) - Prob(D\ (gen(1^k), a{\cdot}P, b{\cdot}P, c{\cdot}P) = 1)| < negl(k), \quad (3)$$

where a, b, and c are randomly chosen in {1, . . . , q}.

Definition 3. *Let gen(1^k) be a parameter generation algorithm that outputs the description of a group G, its generator $P \in G$, and its order q. Let H_0() be a key derivation function. Let $H_1(H_0(ab{\cdot}P,),)$ be a keyed pseudorandom function using the Diffie-Hellman key derivation function if for every probabilistic polynomial time distinguisher D* there exists a negligible function negl such that*

$$|Prob(D^*(gen(1^k), a{\cdot}P, b{\cdot}P, H_1(H_0(ab{\cdot}P,),)) = 1) - Prob(D^*(gen(1^k), a{\cdot}P, b{\cdot}P, R()) = 1)| < negl(k), \quad (4)$$

where a and b are randomly chosen in {1, . . . , q} and R() is a truly random function.

Definition 3 is almost the same as Definition 4 in [31]. However, we replace the Diffie-Hellman key $ab{\cdot}P$ with a key derivation function $H_0(ab{\cdot}P,)$. We have *MacKey* || *LTK* = *f*5(*DHKey*, *Na*, *Nb*, *A*, *B*) in step 5a and step 5b of Π^*, Π', and Π''. Hence, we define *MacKey* = $[f5(DHKey, Na, Nb, A, B)]_{\mathrm{Mac}}$. To reduce our privacy results, the following lemma is required.

Lemma 1. *Prob(* $\mathcal{PG}_{\mathcal{A},\Pi'}[n, k]$ *succeeds in guessing b)* = $\frac{1}{2}$.

Proof of Lemma 1. As shown in Figure 10, we know (*SKa*, *PKa*) and (*SKb*, *PKb*) are updated in each run of Π' and any (*SKx*, *PKx*) in Π' is independent from each other. Hence, during the learning phase of the privacy game $\mathcal{PG}_{\mathcal{A},\Pi'}$, $\mathcal{A}$ learns no knowledge of the bit *b* by calling the Reveal and Corrupt oracles. It means that we can exclude the Reveal and Corrupt oracles in $\mathcal{PG}_{\mathcal{A},\Pi'}$.

In the learning phase of $\mathcal{PG}_{\mathcal{A},\Pi'}$, $\mathcal{A}$ can call the Init, Send, Execute, and Result oracles to any different pairing devices $i, j \in \{1, 2, \ldots, n\}$ and then receive the corresponding responses. Next, $\mathcal{A}$ enters into the challenge phase of $\mathcal{PG}_{\mathcal{A},\Pi'}$. He selects a challenge device $A \in \{1, 2, \ldots, n\}$ and any other device $B \in \{1, 2, \ldots, n\}$ and calls the oracle Test (*A*, *B*). We know that Test oracle acts as follows.

When $b = 0$, it calls Execute (*A*, *B*) and returns its output *PKa*, *PKb*, *Na*, *Nb*, *Ea*, and *Eb* to $\mathcal{A}$. Here, according to the specification of Π', *PKa*, *PKb*, *Na*, *Nb*, *Ea*, and *Eb* are respectively generated by the device *A* and the device *B*.

When $b = 1$, the Test oracle directly generates its *PKa*, *PKb*, *Na*, *Nb*, *Ea*, and *Eb* according to the specification of Π' and returns them to $\mathcal{A}$.

Whether the random bit *b* is 0 or 1, $\mathcal{A}$ always receives a uniform transcript of Π', i.e., *PKa*, *PKb*, *Na*, *Nb*, *Ea*, and *Eb*. Hence, we have:

$$Prob(\mathcal{PG}_{\mathcal{A},\Pi'}[n,\ k] \text{ succeeds in guessing } b) = \frac{1}{2}. \quad (5)$$

This completes the proof. □

Theorem 1. *Assume that P256() in both* Π' *and* Π'' *satisfies the DDH assumption as in Definition 2. Assume that* $[f5()]_{Mac}$ *and f6() in* Π' *and* Π'' *are respectively the key derivation function and the keyed pseudorandom* *function using the Diffie-Hellman key derivation function as in Definition 3. Assume that the device always keeps its pairing information table secret. Then,* (Π', Π'') *is private according to Definition 1.*

Proof of Theorem 1. We know that the Corrupt oracle cannot help $\mathcal{A}$ to guess the random bit b in the Test oracle of the privacy game $\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}$, because all *DHKeys* generated by Π' are random and independent, the pairing information table is kept secret, and the oracles Corrupt ($\Pi^{A,B}$) and Corrupt ($\Pi^{B,A}$) are not allowed if the corresponding *DHKey* is used during the challenge phase of $\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}$. Therefore, we exclude the Corrupt oracle in the following discussions.

To violate the privacy of (Π', Π''), $\mathcal{A}$ completely relies on the correct guess of the random bit b after the Test oracle is called during the challenge phase. When the Test oracle calls Execute (A, B), two devices A and B run either Π' or Π'' if $b = 0$. There are therefore two cases to consider:

Case 1: Π' is run in the Test oracle when $b = 0$. This case means that both the device A and the device B do not maintain the shared *DHKey* in their pairing information tables. When $b = 0$, the Test oracle calls Execute (A, B) and returns the output of Execute (A, B) to $\mathcal{A}$. Here, Execute (A, B) outputs a newly exchanged transcript of Π'. When $b = 1$, the Test oracle also simulates Π' and returns its random transcript of Π' to $\mathcal{A}$. Let E_1 be the event that $\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}$ [n, k] succeeds in guessing b, if Π' is run in the Test oracle when $b = 0$. By Lemma 1, we have $Prob(E_1)$ = $Prob($ $\mathcal{PG}_{\mathcal{A},\Pi'}$[$n$, k] succeeds in guessing b) = $\frac{1}{2}$.

Case 2: Π'' is run in the Test oracle when $b = 0$. This case means that both the device A and the device B maintain the shared *DHKey* in their pairing information tables. During the learning phase of $\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}$, $\mathcal{A}$ can collect and verify the transcripts of (Π', Π''), send the fabricated messages, and disclose the secret keys by calling the Execute, Result, Init, Send, and Reveal oracles. Certainly, $\mathcal{A}$ can implement these operations for the device A and the device B. And then, during the challenge phase of $\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}$, $\mathcal{A}$ must strictly depend on correctly guessing the random bit b in the oracle Test (A, B). Let E_2 be the event that $\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}$ [n, k] succeeds in guessing b, if Π'' is run in the Test oracle when $b = 0$. If (Π', Π'') is private under $\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}$ [n, k], we know that:

$$Prob(E_2) < \frac{1}{2} + negl(k). \tag{6}$$

Let $\delta_1(k)$ be a function such that

$$Prob(E_2) < \frac{1}{2} + \delta_1(k). \tag{7}$$

We demonstrate that $\delta_1(k)$ is negligible by presenting a DDH problem distinguisher D_1 with the same advantage $\delta_1(k)$. The distinguisher D_1 receives ($a{\cdot}P$, $b{\cdot}P$, K) and attempts to determine whether $K = ab{\cdot}P$ or K is a random element in the group G. The distinguisher D_1 simulates the learning phase and the challenge phase of $\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}$ in the following:

Simulation of the learning phase. When $\mathcal{A}$ calls the Execute, Result, Init, Send, and Reveal oracles, the distinguisher D_1 also invokes these oracles to the corresponding devices and returns the outputs of these oracles to $\mathcal{A}$.

Simulation of the challenge phase. When $\mathcal{A}$ selects the device A and the device B and calls the oracle Test (A, B), the distinguisher D_1 simulates the Test oracle as follows:

The distinguisher D_1 flips a fair random coin b.

If $b = 0$, the distinguisher D_1 calls Execute (A, B). It means that a complete run between the partnered instances $\Pi''^{A,B}$ and $\Pi''^{B,A}$ is performed. The output of the Execute oracle is the complete series of messages exchanged by the partnered instances $\Pi''^{A,B}$ and $\Pi''^{B,A}$, i.e., *PKa*, *PKb*, *Na*, *Nb*, *Ea*, and *Eb*. Note that both the device A and the device B obtain the shared *DHKey* in their pairing information tables and the shared *DHKey* has no relation with *PKa* and *PKb*. The distinguisher D_1 returns *PKa*, *PKb*, *Na*, *Nb*, *Ea*, and *Eb* to $\mathcal{A}$.

If b = 1, the distinguisher D_1 sets $PKa = a{\cdot}P$ and $PKb = b{\cdot}P$, randomly generates Na and Nb, and computes $MacKey \parallel LTK = f5(K, Na, Nb, A, B)$, $Ea = f6(MacKey, Na, Nb, rb, IOCapA, A, B)$, and $Eb = f6(MacKey, Nb, Na, ra, IOCapB, B, A)$. The distinguisher D_1 returns its PKa, PKb, Na, Nb, Ea, and Eb to $\mathcal{A}$.

After the query of the Test oracle, the distinguisher D_1 outputs 1 when $\mathcal{A}$ outputs his guessing $b' = b$ in $\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}$. In the view of $\mathcal{A}$, the above simulation by the distinguisher D_1 is exactly $\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}$ when $K = ab{\cdot}P$. Hence, we know:

$$Prob(D_1(\text{gen}(1^k),\ a{\cdot}P,\ b{\cdot}P,\ \text{ab}{\cdot}P)\ =\ 1) < \frac{1}{2} + \delta_1(k). \tag{8}$$

When K is a random element in the group G, $MacKey$ is computed by using the random element K instead of $DHKey = ab{\cdot}P$ and Ea and Eb are further calculated by using $MacKey$. We know that $\mathcal{A}$ cannot guess the random bit b from PKa and PKb, because the device A, the device B, and the distinguisher D_1 all does not use PKa and PKb to generate $DHKey$ in this situation. Let $\delta_2(k)$ be a function such that $\mathcal{A}$ outputs $b' = b$ with probability $\frac{1}{2} + \delta_2(k)$, where $\delta_2(k)$ is the advantage by exploiting $MacKey$, Ea, and Eb. We show that $\delta_2(k)$ is a negligible function by contradiction. Assume that $\delta_2(k)$ is a non-negligible function. We assume that Ea and Eb are generated by the truly random function $R()$ if the random element K is replaced with $DHKey$. We directly construct the distinguisher D_2 to tell the keyed pseudorandom function $f6([f5(DHKey,\ )]_{\text{Mac}},\ )$ from the truly random function $R()$. And the distinguisher D_2 outputs a correct guess with probability $\frac{1}{2} + \delta_2(k)$ by the same reason as the distinguisher D_1 does in the case where the random element K exists. However, the advantage of the distinguisher D_2 is a non-negligible function $\delta_2(k)$, which is contradiction against Definition 3. Hence, $\delta_2(k)$ is a negligible function. We further have:

$$\begin{aligned}&|Prob(D_1(gen(1^k), a{\cdot}P, b{\cdot}P, ab\ P) = 1) - Prob(D_1(gen(1^k), a{\cdot}P, b{\cdot}P, K) = 1)|\\ &= |\tfrac{1}{2} + \delta_1(k)\ -\ \tfrac{1}{2}\ -\ \delta_2(k)|\ =\ |\ \delta_1(k)\ -\ \delta_2(k)|.\end{aligned} \tag{9}$$

By Definition 2, we know that $\delta_1(k)$ is a negligible function due to the negligible function $\delta_2(k)$. Assume that p' denotes the probability of E_1 occurring and p'' denotes the probability of E_2 occurring. Clearly, it follows that $p' + p'' = 1$. We further have:

$$\begin{aligned}&Prob(\mathcal{PG}_{\mathcal{A},(\Pi',\Pi'')}\text{ succeeds in guessing } b) = Prob(E_1)p' + Prob(E_2)p'' <\\ &\tfrac{1}{2}p' + (\tfrac{1}{2} + \delta_1(k))p''\ =\ \tfrac{1}{2} + \delta_1(k)p''.\end{aligned} \tag{10}$$

Thus, we conclude that (Π', Π'') is private by Definition 1. □

6. Performance Evaluation

In this section, we discuss the implementation efficiency of our improvement due to its LE applications. Our performance evaluation has two levels, i.e., the basic protocol analysis and the Bluetooth pairing system analysis. We know that it is not easy to measure the costs of the human operations in the association models. Hence, we omit the costs of these models in all pairing protocols. As shown in Figures 3, 10 and 11, the costs of both pairing devices are all the same in these protocols, if we do not consider the association models. Therefore, we always calculate the costs on one device. The experiments are executed on the following platform:

(1) Experiment environment: The basic cryptographic algorithms are executed on Windows 10 64 bits, Intel(R) Core(TM) i7-8700 CPU @3.20 GHz 3.19GHz, and 8.00 GB RAM.

(2) Cryptographic tool: Python 3.6.6 cryptography toolkit PyCryptodemo.

6.1. Basic Protocol Analysis

6.1.1. Computation Cost

Table 3 shows the computation costs of the LESSP protocol and our improved one. The LESSP protocol is further divided into two cases, i.e., the key pair reuse and the key pair update. The experiment results of these pairing protocols are shown in Figure 12. We can see that the computation efficiency of the aggressive LESSP protocol is significantly higher than other protocols.

Table 3. Computation efficiency of pairing protocols.

Protocol	Computation Cost
LESSP protocol with the key pair reuse	$P256$ [1] + $Rand$ [2] + $3AES\text{-}CMAC_{128}$ [3]
LESSP protocol with the key pair update	$GenKP$ [4] + $P256$ + $Rand$ + $3AES\text{-}CMAC_{128}$
Modified LESSP protocol	$GenKP$ + $P256$ + $Rand$ + $3AES\text{-}CMAC_{128}$
Aggressive LESSP protocol	$GenPK$ [5] + $Rand$ + $3AES\text{-}CMAC_{128}$

[1] *P*256 denotes computation cost of ECDH algorithm; [2] *Rand* denotes computation cost of 128-bit pseudo-random number generator; [3] $AES\text{-}CMAC_{128}$ denotes computation cost of one way cryptographic function; [4] *GenKP* denotes computation cost of ECDH key generator; [5] *GenPK* denotes computation cost of random elliptic curve point generator in Π''.

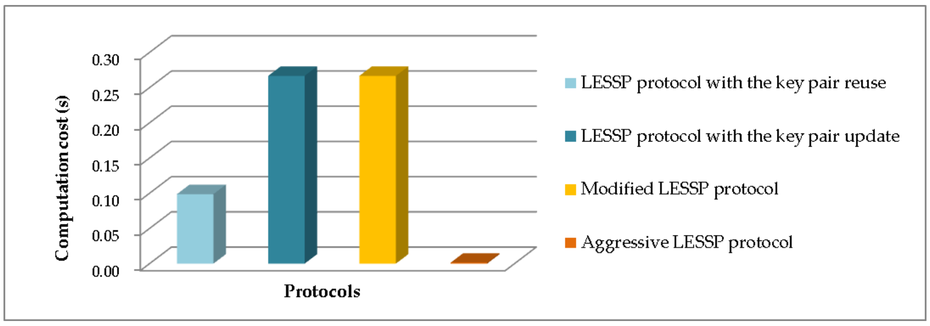


Figure 12. Experiment comparison of computation time costs.

6.1.2. Communication Cost and Storage Cost

As shown in Table 4, we calculate the communication costs of the transmitted messages during the runs of the pairing protocols. And for the storage cost, we merely calculate the long-term parameters, i.e., the reused key pair and the values stored in the pairing information table.

Table 4. Communication and storage costs in one run of each protocol.

Protocol	Communication Cost (bits)	Storage Cost (bits)
LESSP protocol	1024	512
Modified LESSP protocol	1024	$304m$ [1]
Aggressive LESSP protocol	1024	

[1] m denotes the number of the records in the device's pairing information table in Figure 8.

6.1.3. Energy Cost Concern

The energy cost of our improvement is the same as or less than the LESSP protocol, if the cost of searching the pairing information table is free. The reason is that our aggressive LESSP protocol does not need to generate the ECDH private-public key pair and compute the Diffie-Hellman key and other computation and communication costs of both our improvement and the LESSP protocol are the same.

6.2. Bluetooth Pairing System Analysis

We consider the computation costs of a device, which probably pairs with a set of other n^* devices in the Bluetooth pairing system. Let P be the total number of the pairing run and m be the number of the records of the Bluetooth device's pairing information tables in our improvement. Let W be the probability distribution of the devices' pairing runs, where $W = [p_1, p_2, \ldots, p_{n^*}]$ and $p_{1 \le i \le n^*}$ denotes the probability of the event that the device pairs with the device i in the Bluetooth pairing system. We know that $p_1 + p_2 + \ldots + p_{n^*} = 1$.

6.2.1. Number of Devices

We set $P = 10{,}000$ and $m = 5$ and assume $p_1 + p_2 + p_3 + p_4 + p_5 = 0.8$ and $p_6 + p_7 + \ldots + p_{n^*} = 0.2$. We evaluate the computation costs of the LESSP protocol and our improvement, when the number of devices n^* = 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, and 100, respectively. Figure 13 shows the experimental results of the device's computation time costs. Our improvement yields about 100–400% time reduction against the LESSP protocol with the key pair reuse and the LESSP protocol with the key pair update.

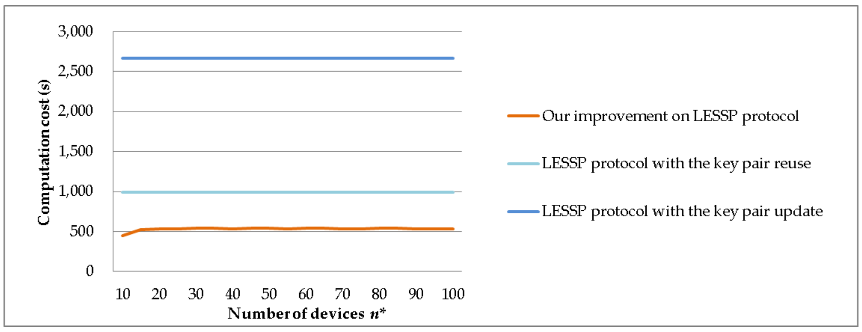


Figure 13. The computation time costs against the number of devices n^*.

6.2.2. Number of Records

We set $n^* = 50$ and $P = 10{,}000$ and assume $p_1 + p_2 + p_3 + p_4 + p_5 = 0.8$ and $p_6 + p_7 + \ldots + p_{50} = 0.2$. We evaluate the computation costs of the LESSP protocol and our improvement, when the number of records m ranges from 5 to 20. Figure 14 shows the experimental results of the device's computation time costs. In Figure 14, we can observe that the time saving of our improvement is similar to the case of the variable number of devices in Section 6.2.1.

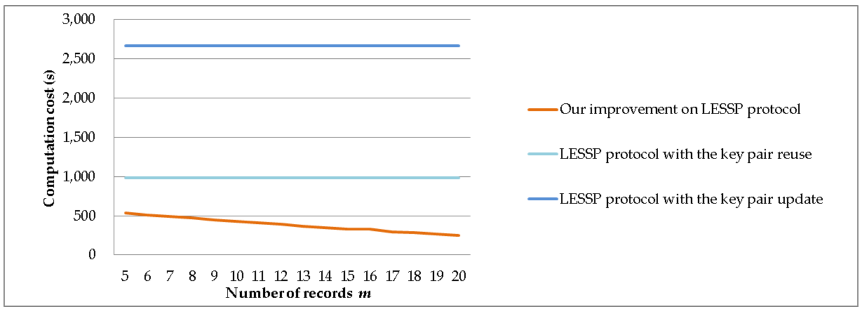


Figure 14. The computation time costs against the number of records m.

6.2.3. Probability Distribution

We set $n^* = 50$, $P = 10{,}000$, and $m = 5$. We evaluate the computation costs of the LESSP protocol and our improvement, when the probability distribution $W = [p_1, p_2, \ldots, p_{50}]$ varies in our improvement. The value range of $p_1 + p_2 + p_3 + p_4 + p_5$ changes from 0.1 to 0.9. Figure 15 shows the device's experimental time costs. When the value of $p_1 + p_2 + p_3 + p_4 + p_5$ increases, the time cost of our improvement significantly reduces. In fact, in wireless personal area networks, it is common for a few devices to have very high pairing probabilities.

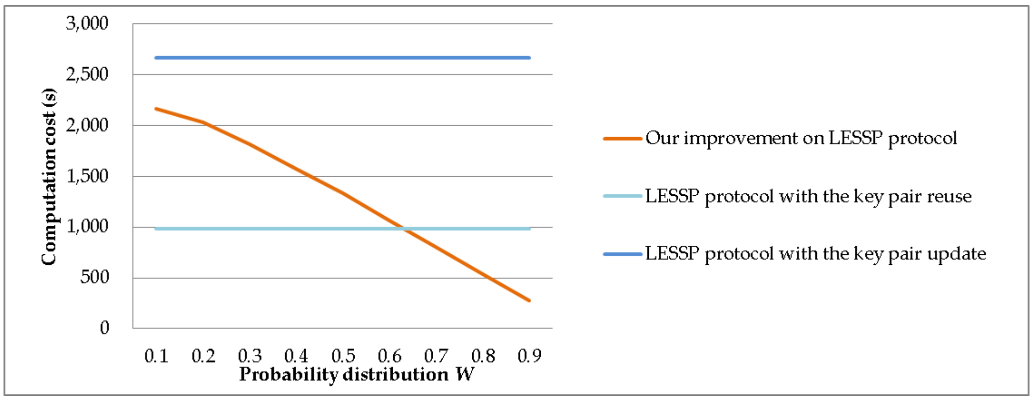


Figure 15. The computation time costs against probability distribution *W*.

We believe that the execution performance of both our improvement and the LESSP protocol should be similar as that observed in Figures 13–15, when they are realized on commercial-off-the-shelf (COTS) LE Bluetooth devices. That is to say, although COTS Bluetooth LE devices may have different processing power, the execution performance is dominated by the corresponding parameters in Sections 6.2.1–6.2.3, whereas COTS Bluetooth LE devices merely determine the detailed values of the computation costs.

7. Conclusions and Future Work

When the LESSP protocol is deployed in the Bluetooth system, an adversary can track the targeted device by its reuse of the private-public key pair. In fact, the SSP protocol has a similar privacy weakness. Our improvement on the LESSP protocol is proven to be able to overcome the privacy weakness under the proposed privacy model. Moreover, the experimental results show that our improvement is as efficient as the LESSP protocol.

It needs to be pointed out that our improvement has not been evaluated with open-source software stacks and software-defined radios. Therefore, our future work is to use the Universal Software Radio Peripheral (USRP) [33] to implement the LESSP protocol and our improvement. Based on it, we can obtain the energy cost and the more accurate efficiency results for these protocols. We conclude with two interesting open problems for the privacy of the LESSP protocol. Firstly, our work has not dealt with the privacy of the association models in the LESSP protocol. In fact, the adversary may violate the privacy of the LESSP protocol by intercepting the communications during the run of the association models. Secondly, we have not investigated the forward/backward privacy of the LESSP protocol. Backward privacy means that, if the adversary reveals the secret of a Bluetooth device A at time τ, the adversary is not able to tell whether a run of the protocol before time τ involves the device A. Forward privacy deals with the adversary who, even with the knowledge of the secret of the device A at time τ, cannot determine whether the same device was involved in a run of the protocol that occurred at time $\tau + \tau'$ (for some $\tau' > 0$). To solve these two privacy problems, we need a new privacy model and a redesigned LESSP protocol.

Author Contributions: Conceptualization, D.-Z.S.; Methodology, D.-Z.S.; Validation, D.-Z.S. and Y.Y.; Formal Analysis, D.-Z.S.; Investigation, L.S.; Writing—Original Draft Preparation, D.-Z.S. and L.S.; Writing—Review and Editing, D.-Z.S. and L.S.; Supervision, D.-Z.S. and Y.Y.; Funding Acquisition, D.-Z.S. and Y.Y.

Funding: This research was funded by the National Science Foundation of China, Grant Nos. 61872264 and 61802276.

Acknowledgments: The authors would like to thank the editor and the reviewers for their valuable suggestions and comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Specification of the Bluetooth System, Covered Core Package Version: 4.0, Master Table of Contents & Compliance Requirements, Bluetooth SIG Proprietary, July 2010. Available online: https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=456433 (accessed on 3 July 2019).
2. Specification of the Bluetooth System, Covered Core Package Version: 4.2, Master Table of Contents & Compliance Requirements, Bluetooth SIG Proprietary, December 2014. Available online: https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=441541 (accessed on 3 July 2019).
3. Specification of the Bluetooth System, Covered Core Package Version: 5.0, Master Table of Contents & Compliance Requirements, Bluetooth SIG Proprietary, December 2016. Available online: https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=421043 (accessed on 3 July 2019).
4. Bluetooth 5 Learning. Available online: https://www.silabs.com/products/wireless/learning-center/bluetooth/bluetooth-5 (accessed on 3 July 2019).
5. Specification of the Bluetooth System, Covered Core Package Version: 5.1, Master Table of Contents & Compliance Requirements, Bluetooth SIG Proprietary, January 2019. Available online: https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=457080 (accessed on 3 July 2019).
6. Specification of the Bluetooth System, Covered Core Package Version: 4.1, Master Table of Contents & Compliance Requirements, Bluetooth SIG Proprietary, December 2013. Available online: https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=282159 (accessed on 15 July 2019).
7. Padgette, J.; Scarfone, K.; Chen, L. *Guide to Bluetooth Security: Recommendations of the National Institute of Standards and Technology*; Special Publication 800-121 Revision 1; CreateSpace Independent Publishing Platform: Scotts Valley, CA, USA, 2012; pp. 1–39.
8. Gomez, C.; Oller, J.; Paradells, J. Overview and evaluation of Bluetooth low energy: An emerging low-power wireless technology. *Sensors* **2012**, *12*, 11734–11753. [CrossRef]
9. Gibbs, J. BLE and Laird's BL6x0 Series & BT900 Modules: A Guide to Security and Privacy. Available online: https://davidhoglund.typepad.com/files/ble__bl600_security_and_privacy_white_paper_final.pdf (accessed on 18 July 2019).
10. Wang, P. Bluetooth Low Energy—Privacy Enhancement for Advertisement. Ph.D. Thesis, Norwegian University of Science and Technology, Trondheim, Norway, June 2014.
11. Raza, S.; Prasant, M.; He, Z.; Voigt, T. Bluetooth smart: An enabling technology for the internet of things. In Proceedings of the 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Abu Dhabi, UAE, 19–21 October 2015; pp. 155–162.
12. Fawaz, K.; Kim, K.H.; Shin, K.G. Protecting privacy of BLE device users. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 1205–1221.
13. AN99209. PSoC®4 BLE and PRoC™ BLE: Bluetooth LE 4.2 Features. Available online: https://www.cypress.com/file/224826/download (accessed on 18 July 2019).
14. Hassan, S.S.; Bibon, S.D.; Hossain, M.S.; Atiquzzaman, M. Security threats in Bluetooth technology. *Comput. Secur.* **2018**, *74*, 308–322. [CrossRef]
15. Mare, S.; Sorber, J.; Shin, M.; Cornelius, C.; Kotz, D. Hide-n-sense preserving privacy efficiently in wireless mHealth. *Mob. Netw. Appl.* **2014**, *19*, 331–344. [CrossRef]
16. Cyr, B.; Horn, W.; Miao, D.; Specter, M. Security Analysis of Wearable Fitness Devices (Fitbit). Available online: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2015/03/20082016/17-cyrbritt-webbhorn-specter-dmiao-hacking-fitbit.pdf (accessed on 18 July 2019).
17. Das, A.K.; Pathak, P.H.; Chuah, D.N.; Mohapatra, P. Uncovering privacy leakage in BLE network traffic of wearable fitness trackers. In Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications (HotMobile 2016), St. Augustine, FL, USA, 23–24 February 2016; pp. 99–104.
18. Korolova, A.; Sharma, V. Cross-app tracking via nearby Bluetooth low energy devices. In Proceedings of the 8th ACM Conference on Data and Application Security and Privacy (CODASPY '18), Tempe, AZ, USA, 19–21 March 2018; pp. 43–52.
19. Sivakumaran, P.; Blasco, J. A study of the feasibility of co-located app attacks against BLE and a large-scale analysis of the current application-layer security landscape. *arXiv* **2018**, arXiv:1808.03778.

20. Cha, S.C.; Chuang, M.S.; Yeh, K.H.; Huang, Z.J.; Su, C. A user-friendly privacy framework for users to achieve consents with nearby BLE devices. *IEEE Access* **2018**, *6*, 20779–20787. [CrossRef]
21. Sandhya, S.; Devi, K.A.S. Analysis of Bluetooth threats and v4.0 security features. In Proceedings of the 2012 International Conference on Computing, Communication and Applications, Dindigul, India, 22–24 February 2012; pp. 1–4.
22. Phan, C.W.; Mingard, P. Analyzing the secure simple pairing in Bluetooth v4.0. *Wirel. Pers. Commun.* **2012**, *64*, 719–737. [CrossRef]
23. Zegeye, W.K. Exploiting Bluetooth low energy pairing vulnerability in telemedicine. In Proceedings of the 50th Annual International Telemetering Conference, Las Vegas, NV, USA, 26–29 October 2015; pp. 1–10.
24. Gajbhiye, S.; Karmakar, S.; Sharma, M.; Sharma, S. Two-party secure connection in Bluetooth-enabled devices. *Inf. Secur. J.* **2018**, *27*, 42–56. [CrossRef]
25. Ha, J.H.; Moon, S.J.; Zhou, J.; Ha, J.C. A new formal proof model for RFID location privacy. In Proceedings of the 13th European Symposium on Research in Computer Security (ESORICS 2008), Málaga, Spain, 6–8 October 2008; pp. 267–281.
26. Juels, A.; Weis, S.A. Defining Strong Privacy for RFID. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2009**, *13*, 1–23. [CrossRef]
27. Kostakos, V. The privacy implications of Bluetooth. *arXiv* **2008**, arXiv:0804.3752.
28. Lindell, A.Y. Comparison-based key exchange and the security of the numeric comparison mode in Bluetooth v2.1. In Proceedings of the Cryptographers' Track at the RSA Conference 2009 (CT-RSA 2009), San Francisco, CA, USA, 20–24 April 2009; pp. 66–83.
29. Sun, D.Z.; Li, X.H. Vulnerability and enhancement on Bluetooth pairing and link key generation scheme for security modes 2 and 3. In Proceedings of the Information and Communications Security: 18th International Conference (ICICS 2016), Singapore, 29 November–2 December 2016; pp. 403–417.
30. Sun, D.Z.; Mu, Y.; Susilo, W. Man-in-the-middle attacks on secure simple pairing in Bluetooth standard V5.0 and its countermeasure. *Pers. Ubiquit. Comput.* **2017**, *22*, 55–67. [CrossRef]
31. Sun, D.Z.; Sun, L. On Secure Simple Pairing in Bluetooth Standard v5.0-Part I: Authenticated Link Key Security and Its Home Automation and Entertainment Applications. *Sensors* **2019**, *19*, 1158. [CrossRef] [PubMed]
32. Katz, J.; Lindell, Y. *Introduction to Modern Cryptography: Principles and Protocols*, 1st ed.; Chapman & Hall/CRC: Boca Raton, FL, USA, 2007; pp. 441–452.
33. Ettus Research LLC. Available online: http://www.ettus.com (accessed on 3 July 2019).