mathematics

MDPI

Article

Address Privacy of Bluetooth Low Energy

Dazhi Sun [1,*] and Yangguang Tian [2]

[1] Tianjin Key Laboratory of Advanced Networking (TANK), College of Intelligence and Computing, Tianjin University, Tianjin 300350, China
[2] Department of Computer Science, University of Surrey, Surrey GU2 7XH, UK
* Correspondence: sundazhi@tju.edu.cn; Tel.: +86-22-2740-1091

Abstract: Bluetooth low energy (LE) devices have been widely used in the Internet of Things (IoT) and wireless personal area networks (WPAN). However, attackers may compromise user privacy by tracking the addresses of the LE device. The resolvable private address (RPA) mechanism provides address privacy protection for the LE device. Similar to Zhang and Lin's work in CCS 2022, we investigate the privacy of the RPA mechanism in this paper. Our contributions are threefold. First, we discover that the RPA mechanism has a privacy weakness. The attacker can track the targeted device by exploiting the runs of the RPA mechanism when he intercepts the targeted device's obsolete RPA value. Second, we propose an improved RPA mechanism to overcome the privacy weakness in the RPA mechanism. The improved RPA mechanism leads to a small amount of extra overheads without requiring modification to the basic cryptographic tools used in the standard specification. Third, we formalize a privacy model to capture the address privacy of the RPA mechanisms. Our improved RPA mechanism provides enhanced privacy guarantees to Bluetooth LE devices in wireless personal applications.

Keywords: Bluetooth standard; low energy; resolvable private address mechanism; traceability; privacy; cryptography

MSC: 68M12

Citation: Sun, D.; Tian, Y. Address Privacy of Bluetooth Low Energy. *Mathematics* **2022**, *10*, 4346. https://doi.org/10.3390/math10224346

Academic Editor: Antanas Cenys

Received: 11 October 2022
Accepted: 17 November 2022
Published: 19 November 2022

1. Introduction

Bluetooth is an open technology standard for short-range radio frequency communication. Bluetooth hardware and software modules are widely used in various kinds of consumer and business devices including mobile phones, headsets, laptops, keyboards, mice, tablets, and automobiles. Power consumption is a crucial but challenging factor, when the wireless and mobile devices deploy Bluetooth hardware and software modules. As the technical innovation, the Bluetooth low energy (LE) therefore aims to support low-power and low-cost wireless communications. In practice, Bluetooth LE offers a highly efficient approach to build the Internet of Things (IoT) and wireless personal area networks (WPAN) [1].

LE devices may compromise user privacy through their Bluetooth transmitted packets. As shown in Figure 1, the attacker employs a sniffer to intercept and analyze the transmitted packets among LE devices. If a user bonds with a device in some applications such as [2,3], then the device's transmitted packets will potentially disclose the user identity. Bluetooth standard specifications [4–6] therefore provide a privacy solution for the hostile environments [7–10]. The privacy solution should ensure that the attacker cannot exploit the transmitted packets to identify the targeted device and determine the owner of it.

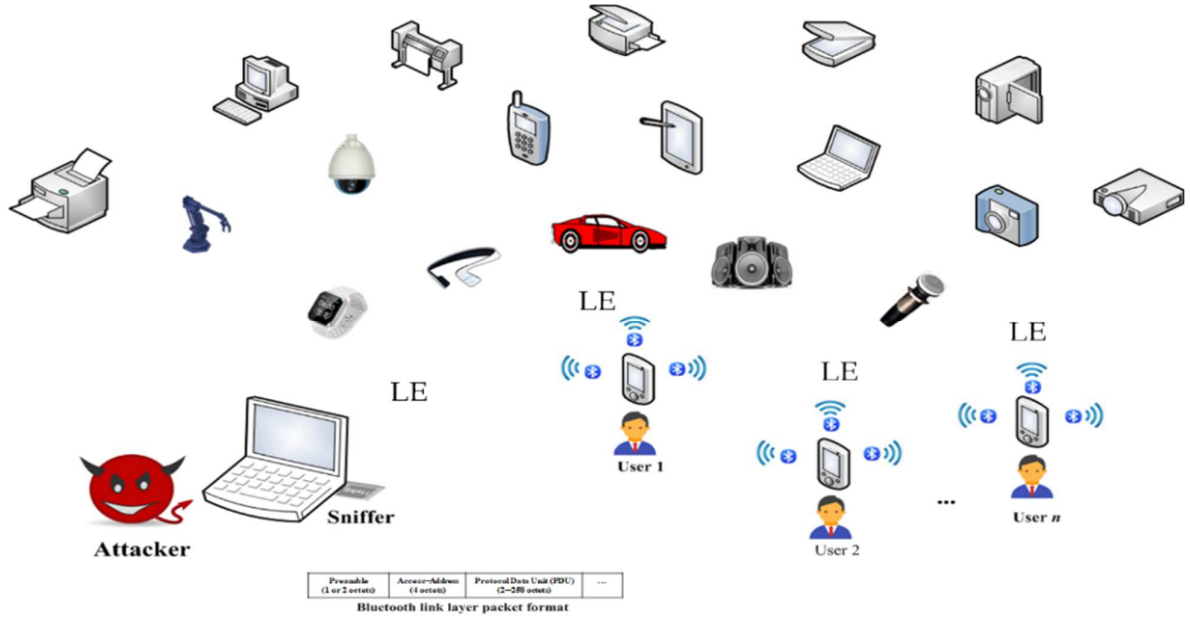

Figure 1. Privacy analysis of Bluetooth personal communication.

1.1. Previous Work on Bluetooth Privacy

Padgette et al. [11] introduced the privacy capabilities of Bluetooth technology. Càsar et al. [12] specially surveyed the privacy vulnerabilities and their countermeasures for the different LE versions. We further summarize the research work of the LE privacy.

- *Privacy of advertisement and device address.* Some literature [13,14] focused on implementing the addresses privacy mechanisms for the LE devices. Some literature [15–20] showed that the advertising procedure possibly leaks the identity information of the LE devices and therefore enhances the privacy of the advertising procedure. Ludant et al. [21] reported that LE advertisements could link to Bluetooth classic frames and the device's globally unique identifier (i.e., BDADDR) due to the bad design of Bluetooth chips. They also developed several mitigations for the Bluetooth stack. Very recently, regarding Bluetooth LE, Zhang and Lin [22] showed that the address randomization scheme using the message authentication code (MAC) is vulnerable to replay attacks and further suggested timestamps-based randomized MAC addresses.
- *Privacy of secure connection.* Secure connection is the basis for LE devices to achieve authentication, integrity, confidentiality and other security services. The task of secure connection is to establish the link key between devices. In [23], we demonstrated the privacy vulnerability of the secure connection due to the reuse of the Diffie–Hellman key, and enhanced the privacy of the secure connection. Zhang et al. [24] showed downgrade attacks on secure connections only (SCO) and built a prototype for the SCO mode on Android 8 atop Android open-source project (AOSP). Tschirschnitz et al. [25] described a design flaw in the pairing mechanism of Bluetooth called method confusion and proposed changes to the Bluetooth specification that immunize it against method confusion.

In addition, many researchers proposed Bluetooth privacy systems [26–28] at the application level to protect the user privacy. Due to the fast development of Bluetooth WPAN, we can see that more and more privacy protection features have been included in the newest Bluetooth standard [6].

1.2. Bluetooth Address and Its Privacy

Bluetooth standard applies the address mechanism to identify each device and its transmitted packets, which is analogous to TCP/IP network. To establish a Bluetooth

connection, a device must transmit its Bluetooth address to another remote device. Hence, to compromise user privacy, a practical way is to track the address transmitted by the user's Bluetooth device.

We summarize the type of Bluetooth device addresses in Figure 2. The address may be either a public device address or a random device address, all of which are 48 bits in length. Meanwhile, the device should use at least one type of device addresses and may contain both. Each public device address is a unique 48-bit Bluetooth address in accordance with Section 8.2 universal addresses in the IEEE 802-2014 standard [29]. A public device address maintains a 24-bit company identifier and a 24-bit company assigned device identifier. Public device address as the MAC address of TCP/IP network is an invariant value for each device. Random device address consists of random static device address and private device address. Private device address is either non-resolvable private address or resolvable private address (RPA) and can be updated according to the period of the timer. In practice, Bluetooth devices rarely use non-resolvable private addresses. An RPA shall be generated by an identity resolving key (IRK) and a random number. In order to reconnect to the known device, the RPA must be resolvable by the peer device. Identity address (IA) is a public device address or random static device address. Each device must maintain an IA. That is, the device shall also have an IA if it is using a private device address.

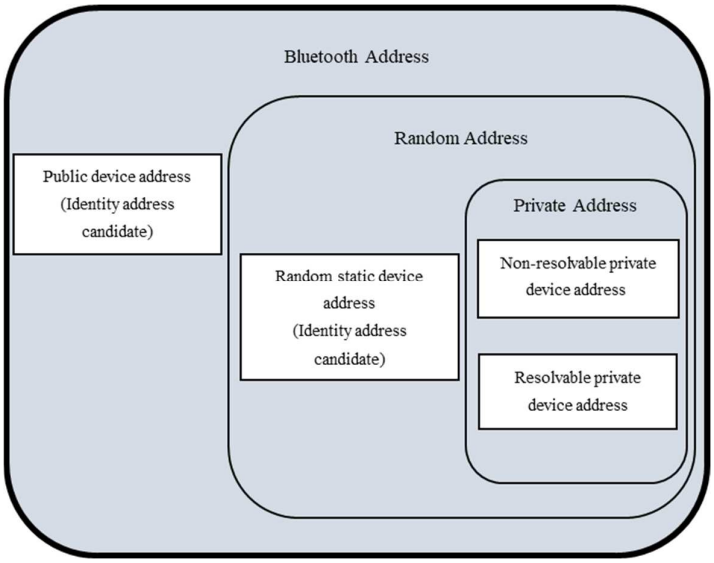

Figure 2. Family of Bluetooth address.

Discussion. Bluetooth basic rate/enhanced data rate devices merely adapt public device address. Comparatively, Bluetooth LE devices support not only public device address but also random device address. The Bluetooth standard of LE newly increases random device address due to the following concerns. First, public device address needs to buy from the IEEE organization. The large number of the low-cost LE devices cannot bear the expenses of public device addresses, even though the price of each address is cheap. Second, the application and management of a public device address is more complex than that of a random device address. Third, LE devices always run on broadcast communication mode. The attacker can easily intercept the transmitted packets, such as the device's address. Hence, the invariant address incurs the security and privacy threats.

1.3. Our Contributions

To protect user privacy, the Bluetooth LE device should employ private device address. Moreover, the RPA mechanism can provide strong privacy protection compared to the non-resolvable private address mechanism, because RPA is generated and resolved by IRK and cryptographic algorithm. However, the attacker could exploit the breaches in the RPA

mechanism to compromise the user privacy. Therefore, we systematically investigate the privacy problem of the RPA mechanism.

Our contributions on the RPA mechanism are threefold. First, we demonstrate that the attacker can track the targeted device by using the runs of the RPA mechanism. That is, the attacker replays the used RPAs to the counterpart of the targeted device and confirms that the targeted device is present by the counterpart's responses. Second, we propose an improved RPA mechanism to overcome the exposed privacy weakness in the RPA mechanism. To avoid replaying the used RPAs, our trick is to maintain the RPA counters in the device. Third, we propose a formal model to evaluate the privacy of the RPA mechanisms. Moreover, our proposed model is a tool to evaluate the privacy of the RPA mechanisms in the future Bluetooth standards.

We notice that Zhang and Lin's recent work [22] is also dedicated to the privacy of the RPA mechanism. However, our work is different from Zhang and Lin's work. We outline the differences as follows:

(1) Although both attacks exploit replaying the sniffed RPA values to probe whether a device will respond or not, Zhang and Lin's attack focuses on the linkage relations among the obsolete RPA values, tracking the targeted device using a real-time tunnel, and tracking the absent device. Our attack aims to recognize the targeted device that currently runs the RPA mechanism.
(2) Zhang and Lin proposed a timestamp-based RPA mechanism and discussed possibility of the synchronized sequence number-based RPA mechanism and storage-based RPA mechanism. Our improvement only employs the counter to prevent the existing attacks. Note that the counter is not a sequence number because it does not require strict synchronization. We argue that the trust timestamp is not easily available in the IoT environments.
(3) We propose a formal model to evaluate the privacy of the RPA mechanisms and further prove that our improvement is private under the proposed privacy model. However, Zhang and Lin's work does not evaluate their timestamp-based RPA mechanism using the provable security approach. In fact, it is impossible due to the timestamp.

In addition, Zhang and Lin's work is practice-oriented and our work is theory-oriented.

2. RPA Mechanism

We assume that A and B are two LE devices and run the RPA mechanism to protect their address privacy. Let IA_A be A's IA and IA_B be B's IA, and let IRK_A be A's IRK and IRK_B be B's IRK.

2.1. Flow of RPA Mechanism

For a self-contained discussion, we review the RPA mechanism and follow the description style of [14].

2.1.1. Initial Connection Procedure

As shown in Figure 3, both A and B firstly implement the initial connection procedure. We explain the detailed flow of the initial connection procedure as follows.

Step 1. *Connectable undirected advertisement*. A broadcasts its advertisement packets, which are not directed at a particular recipient B. The advertisement packets only contain A's IA_A.

Step 2. *Connect request*. This is the request to initiate connection sent from B to A. B responses an advertisement packet, which has B's IA_B with the receiving IA_A. Once A receives this packet, A and B share IA_B and IA_A.

Step3. *Sharing IA and IRK over secure connection*. Both A and B use IA_A and IA_B to run the LE secure connections pairing. After that, both devices exchange their IRK_A, IA_A, IRK_B, and IA_B over the LE secure connection, that is, these data are encrypted by the shared long-term key generated by the LE secure connections pairing. Then, both devices respectively store the peer IRK and IA along with the local IRK in their resolving lists.

Figure 4 depicts the logical structure of the resolving list. In addition, both devices enable address resolution and set their RPA timeouts. Now, both devices have all information necessary to process RPAs. Both devices exchange other packets on the LE data channels after that.

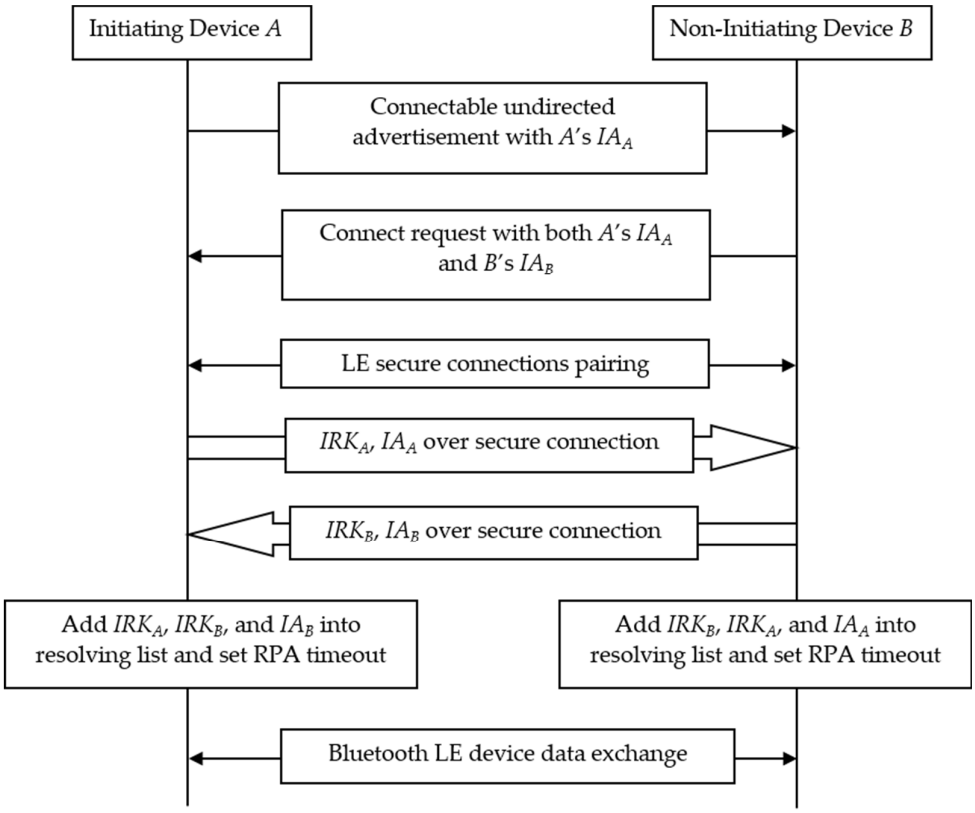

Figure 3. Initial connection procedure.

Resolving List

Local IRK	Peer IRK	Peer IA	Address Type
Local IRK	Peer IRK	Peer IA	Address Type
Local IRK	Peer IRK	Peer IA	Address Type
Local IRK	Peer IRK	Peer IA	Address Type
Local IRK	Peer IRK	Peer IA	Address Type
Local IRK	Peer IRK	Peer IA	Address Type

Figure 4. Logical structure of resolving list.

2.1.2. Reconnection Procedure

Both *A* and *B* need to restore the reconnection, once their initial connection is invalid. To prevent an impersonation attack, the reconnection procedure also should verify the peer IA. As shown in Figure 5, each device identifies the peer by resolving the receiving RPA in accordance with its resolving list. Hence, for each run, both devices can send the random RPAs instead of the invariant IAs to protect their address privacy. We will explain how to generate and resolve RPA in Section 2.2. Let RPA_A be *A*'s RPA and RPA_B be *B*'s RPA. The reconnection flow is in the following.

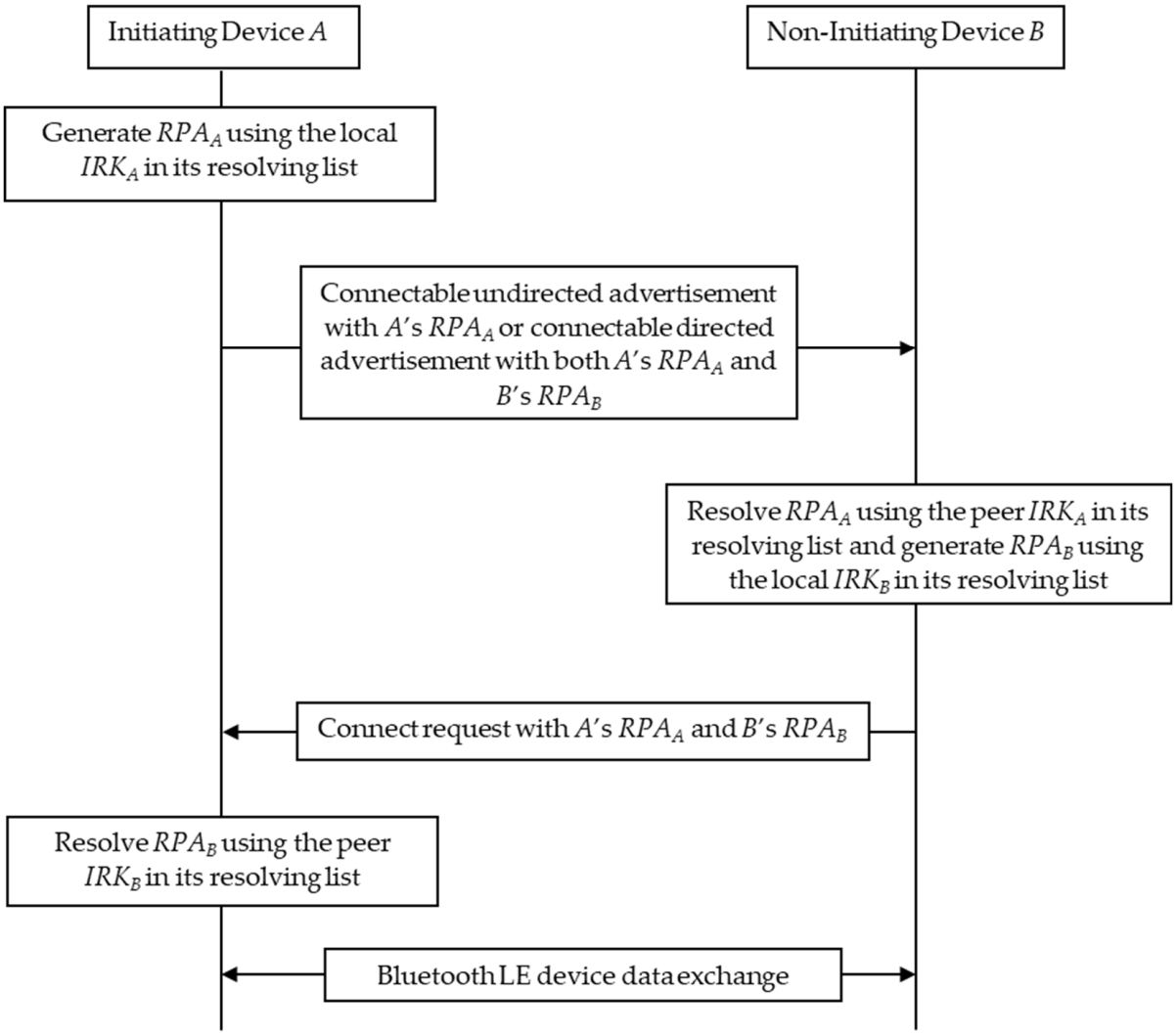

Figure 5. Reconnection procedure using RPA.

Step 1. *Connectable undirected/directed advertisements*. *A* puts the RPA_A into all its transmitted packets and advertises them. Here, *A* allows either connectable undirected advertisements or connectable directed advertisements.

Step 2. *Connect request*. In connect request, *B*'s transmitted packet contains both *A*'s RPA_A and *B*'s RPA_B. RPA_A and RPA_B are shared by *A* and *B* as long as *A* receives *B*'s packet. If RPA_A and RPA_B are correctly verified by *B* and *A*, RPA_A and RPA_B can be used in their subsequent transmitted packets. Both devices exchange the packets in the LE data channels.

2.2. Generation and Resolution of RPA

As shown in Figure 6, RPA consists of 24-bit random number and 24-bit hash value. The random number is known as *prand*, where 22 bits are random, and 2 bits are fixed. To form RPA_A, *A* generates a random *prand* and computes the hash value as follows.

$$Hash = ah(IRK_A, padding||prand) \bmod 2^{24}, \quad (1)$$

where IRK_A is its 128-bit IRK, *padding* is 104-bit zero padding to extend *prand* to 128 bits, *ah* generates 128-bit ciphertext data from a 128-bit key and 128-bit plaintext data using the AES-128-bit block cipher as defined in FIPS-197 [30], and $||$ denotes concatenation operator. RPA_A is formed by

$$RPA_A = hash||prand. \quad (2)$$

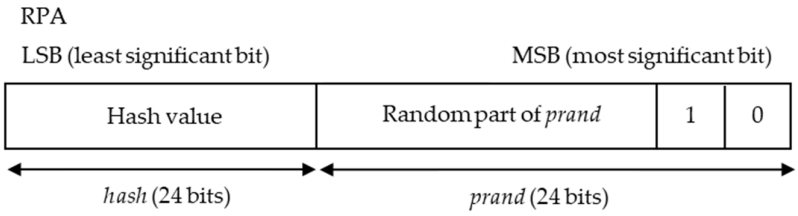

Figure 6. Format of RPA.

To decode the received RPA_A, the least significant 24 bits of RPA_A are extracted into B's *hash*, and the most significant 24 bits of RPA_A are extracted into B's *prand*. Then, B generates its *localhash* by

$$Localhash = ah(IRK_{\mathrm{A}}, padding||prand) \bmod 2^{24}, \quad (3)$$

where IRK_A is from A during the initial connection procedure. B further compares *localhash* with *hash* extracted from RPA_A. If they are equal, B successfully resolves A's RPA_A and derives A's IA_A from its resolving list. Figure 7 depicts RPA resolution for RPA_A. Similarly, B generates its RPA_B and then A can decode it. The device may respond on reception of the next event, if it fails to resolve a private address within time inter frame space.

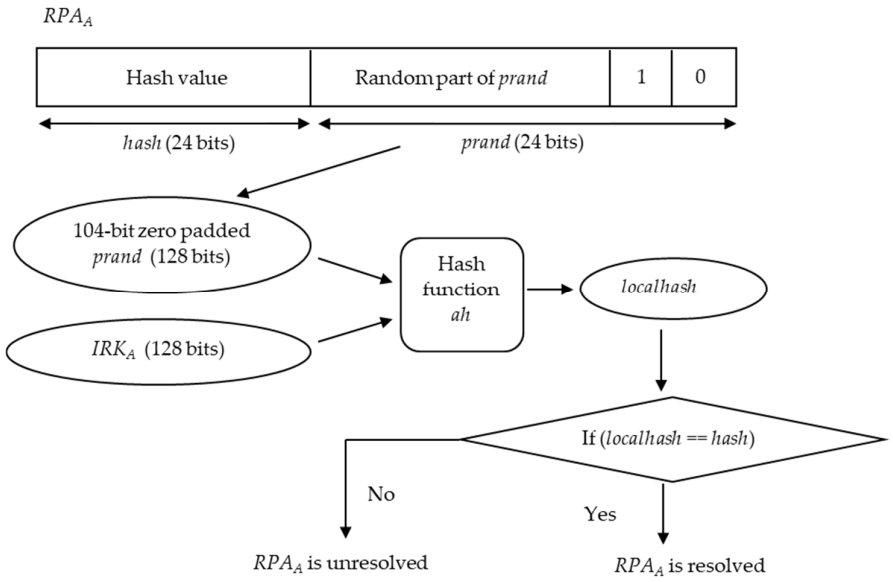

Figure 7. RPA resolution.

3. Privacy Weakness in RPA Mechanism

In the initial connection procedure, the advertisements must directly send both IAs, because the device without the peer IRK is not able to resolve the peer RPA to identify the counterpart. Hence, the attacker can track the device by observing IA in the transmitted packets, if IA is not changed. Comparatively, all transmitted packets in the reconnection procedure are less susceptible to tracking, because both devices employ the random RPAs instead of the invariant IAs for all transmitted packets. Both devices regenerate and resolve their random RPAs for each run of the reconnection procedure. The idea of the RPA mechanism is that the device updates RPA within the specified time interval to enhance the address privacy.

However, we demonstrate that an attacker E can track any device by exploiting the reconnection procedure. Assume that E obtains any targeted A's obsolete but legal RPA $RPA^O{}_A$ with B. E can confirm $RPA^O{}_A$, because he observed that B accepts it. When an unknown device C is running the reconnection procedure with B, E checks the validity of A's $RPA^O{}_A$ to recognize C. Figure 8 shows our attack on the reconnection procedure. We describe it as follows.

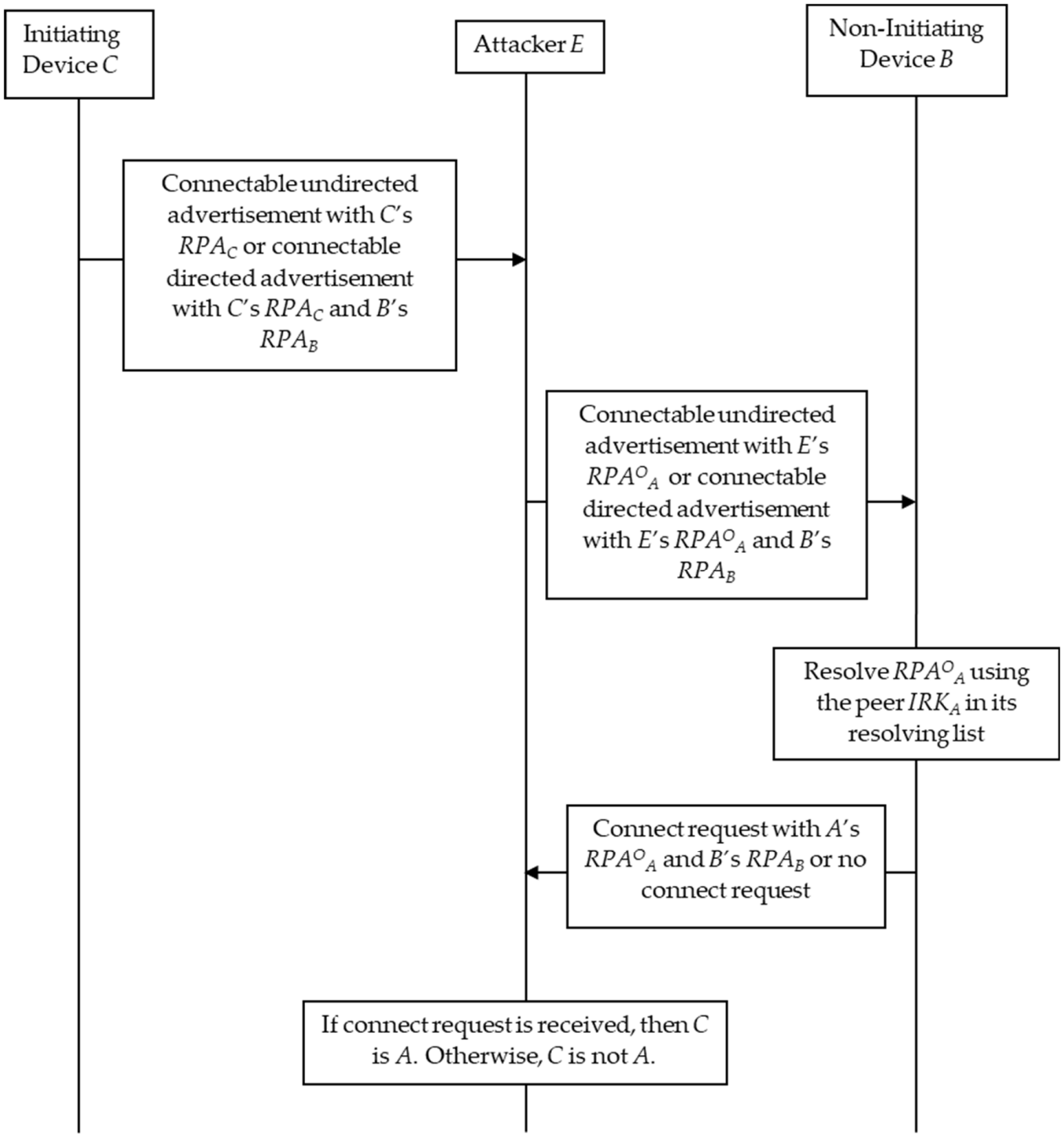

Figure 8. Tracking A using the reconnection procedure.

Step 1. When any device C sends its new RPA RPA_C during connectable undirected advertisements or connectable directed advertisements, E replaces RPA_C by his $RPA^O{}_A$.

Step 2. E waits B's connect request. If E receives the connect request, then E confirms that C is the targeted A; else E believes that C is not the targeted A.

If C is the targeted A, it really generates $RPA^O{}_A$ before. Hence, we know B correctly resolves $RPA^O{}_A$ using IRK_A in its resolving list. As a result, B should send the connect request to E. Otherwise, B does not send connect request, because B is unable to resolve $RPA^O{}_A$. Assume that there are n different device addresses in B's resolving list. Due to the analysis above, the probability that E correctly recognizes A is $1/n$. Clearly, our attack can easily extend to track multiple devices, when E collects the used RPAs of these devices.

In [22], Zhang and Lin suggested tracking a victim's real-time location w/ (or w/o) tunneling. To track the targeted A, their attack demands that E exploits a tunnel, such as the

wormhole attack, to relay A's message to B (or E replays the old $RPA^O{}_A$ after A is absent). In our attack, E blocks the unknown C's current run of the reconnection procedure and determines whether C is the targeted A. Our attack does not require the tunnel compared with the victim's real-time location w/ and has more real-time feature compared with the victim's real-time location w/o.

4. Improved RPA Mechanism

In the reconnection procedure of the RPA mechanism, the devices resolve all the receiving RPAs and never verify and confirm whether they are fresh. Hence, when the attacker replays the used RPAs to the devices, the devices always resolve them and return the resolving results to the attacker. This helps the attacker track the targeted device. In the following, we improve the RPA mechanism to defeat the address tracking due to abuse of the reconnection procedure.

4.1. Improved Initial Connection Procedure

As shown in Figure 9, we redesign the resolving list. We add two new fields for the resolving list, i.e., local RPA counter and peer RPA counter. Local RPA counter is responsible to record the number of RPAs generated by the local device. Similarly, peer RPA counter stores the number of RPAs resolved to the peer device. Let LC_A and PC_A denote A's local RPA counter and peer RPA counter and LC_B and PC_B denote B's local RPA counter and peer RPA counter, and let LC_A, LC_B, PC_A, and PC_B be t bits, where t is a positive integer. It means that IRK_A and IRK_B at most use 2^t times to generate and resolve RPA_A and RPA_B and after that IRK_A and IRK_B need be renewed by both A and B.

Improved Resolving List

Local IRK	Local RPA Counter	Peer IRK	Peer RPA Counter	Peer IA	Address Type
Local IRK	Local RPA Counter	Peer IRK	Peer RPA Counter	Peer IA	Address Type
Local IRK	Local RPA Counter	Peer IRK	Peer RPA Counter	Peer IA	Address Type
Local IRK	Local RPA Counter	Peer IRK	Peer RPA Counter	Peer IA	Address Type
Local IRK	Local RPA Counter	Peer IRK	Peer RPA Counter	Peer IA	Address Type
Local IRK	Local RPA Counter	Peer IRK	Peer RPA Counter	Peer IA	Address Type

Figure 9. Logical structure of improved resolving list.

The improved initial connection procedure is almost the same as the initial connection procedure described in Figure 3. Meanwhile, the improved initial connection procedure demands both A and B, respectively, set LC_A, PC_A, LC_B, and PC_B to 0 during Step 3 of the initial connection procedure. This is the only difference between these two procedures.

4.2. Improved Reconnection Procedure

Figure 10 shows the improved reconnection procedure. Here, we only consider the connectable directed advertisement mode. The improved reconnection procedure keeps the same transmitted packets with that of the reconnection procedure described in Figure 5. The big difference is generation and resolution of devices' RPAs. Moreover, both devices need, respectively, to update LC_A, RC_A, LC_B, and RC_B in their resolving lists. We explain them in the following.

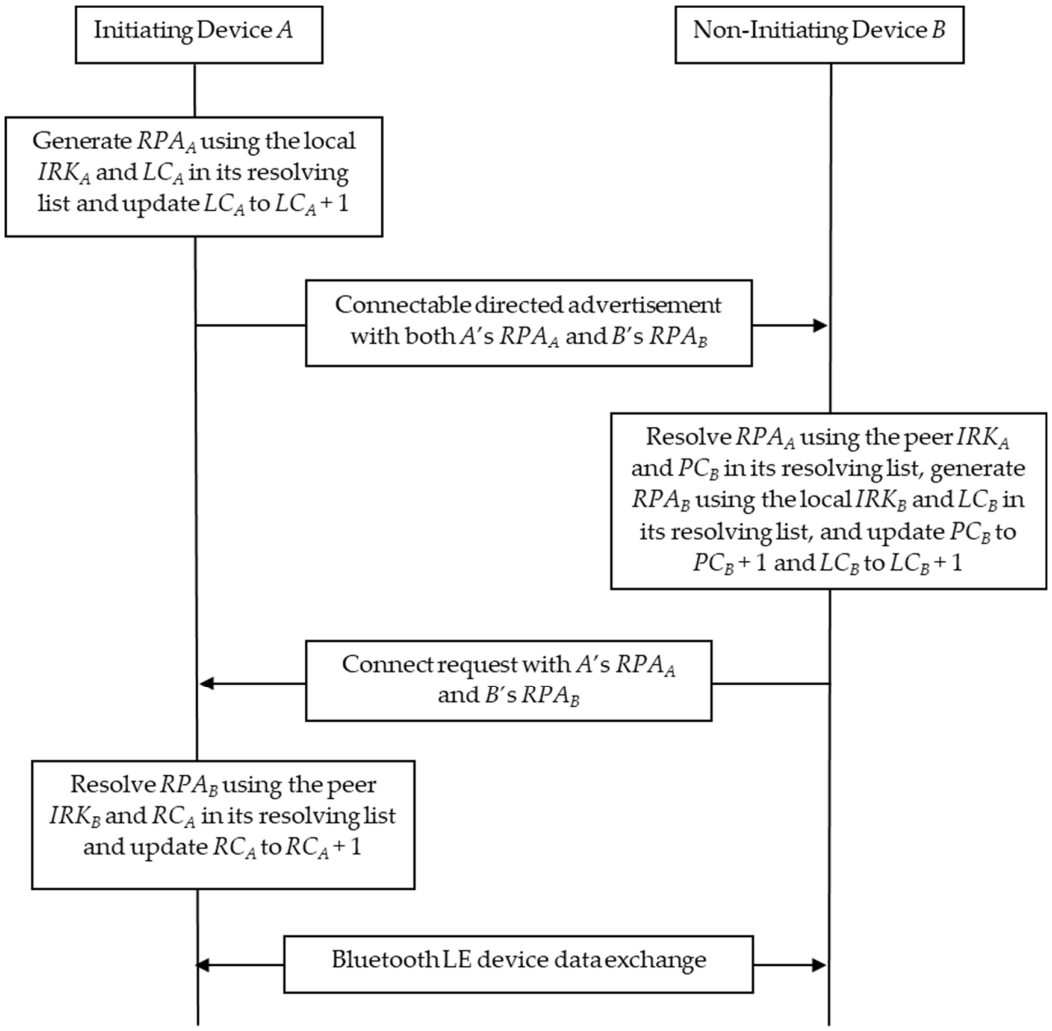

Figure 10. Improved reconnection procedure using RPA.

4.3. Processing RPA/Local RPA Counter/Peer RPA Counter

In the improved RPA mechanism, the format of RPA stays the same with that of the RPA mechanism in Figure 6. To form RPA_A, A generates a random *prand* and computes the value *hash* as follows.

$$Hash = ah(IRK_A, padding1 || LC_A || prand) \bmod 2^{24}, \tag{4}$$

where LC_A is A's t-bit local RPA counter and $padding1$ is $(104-t)$-bit zero. The symbols IRK_A and *prand* and the function *ah* are same as the RPA mechanism. RPA_A still is

$$RPA_A = hash || prand. \tag{5}$$

To decode the received RPA_A, B extracts the least significant 24 bits of RPA_A as its *hash* and the most significant 24 bits of RPA_A as its *prand*. Next, B computes its *localhash* by using

$$localhash = ah(IRK_A, padding1 || RC_B || prand) \bmod 2^{24}, \tag{6}$$

where RC_B is the peer RPA counter in its resolving list. Now, B compares *localhash* with *hash* extracted from RPA_A. If *localhash* is equal to *hash*, B successfully resolves RPA_A and identifies A by IA_A in its resolving list and sets $PC_B = PC_B + 1$. Figure 11 shows the resolution process of RPA_A. The format, generation, and resolution of RPA_B are fully same as these of RPA_A.

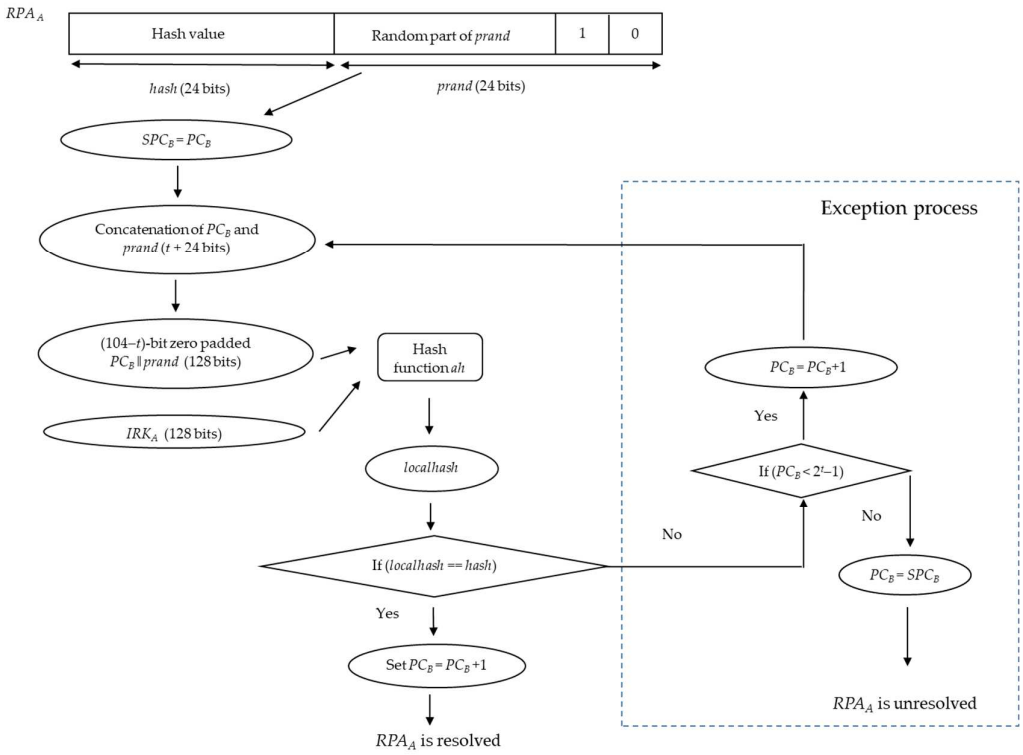

Figure 11. Improved RPA resolution.

However, the attackers can launch a desynchronization attack, that is, he blocks a few messages from A to make the counter out of synchronization. In this situation, A's LC_A and B's PC_B may desynchronize, that is, $LC_A > PC_B$. We argue that this desynchronization attack does not violate the privacy of A, because B has no response to the counterpart when A uses the desynchronized LC_A to generate RPA_A. On the debit side, the desynchronization attack leads B to fail to identify A.

The improved RPA resolution therefore needs to process this exception. The device executes the exception operation (see Figure 11), when all RPAs in the resolving list are unable to solve. If *localhash* is not equal to *hash*, B is required to setting $PC_B = PC_B + 1$ and repeating the decoding operation until $PC_B = 2^t - 1$. If B is unable to resolve RPA_A and $PC_B = 2^t - 1$, it indicates that RPA_A is not generated by A, and PC_B should be restored to its original value. Hence, the variable SPC_B stores the original value of PC_B. When the unsuccessful resolution happens, B sets $PC_B = SPC_B$. This exception operation does not violate the privacy of A if those unsynchronized RPA values are not replayed, because B has no response unless the synchronization of LC_A and PC_B is recovered. That is, B gives a response only if the RPA value is never verified before. We advise t = 6~10. It means that the device can generate 64~1024 different RPA values for the reconnection procedure. At the same time, the device requires at most 64~1024 *ah* computations for each record in its resolving list when the desynchronization attack happens. Our suggestion of the parameter t is practical, when the number of records in its resolving list are not too large, for example, the number of records is 10~20. To defend the desynchronization attack, another alternative countermeasure is to require the update of IRK and resetting resolving lists between the pairing devices, if the desynchronization attack takes place.

4.4. Perforamce Evaluation of Our Improvement

We first analyze the time complexity of the improved RPA mechanism and compare this index between the improved RPA mechanism and the RPA mechanism. Here, we omit to consider the time complexity of initial connection procedure, because initial connection procedures of both mechanisms are the same and involve the LE secure connections pairing.

Assume that each device on average resolves f RPAs using its resolving list to finish reconnection operation. In the case of the counters synchronization, the device requires computing f times of hash function *ah* during our improved reconnection procedure, where f is a positive integer. In the case of the counters desynchronization, the device at most needs to compute $f2^t$ times of hash function *ah* during our improved reconnection procedure. The device comparatively requires f times of hash function *ah* during the reconnection procedure. We know that the time overheads of counter operations and parsing the *prand* can be negligible, compared with the hash computation. Therefore, on average, the time complexity factor with the counters synchronization is

$$w_{cs} = \frac{number\ of\ improved\ procedure's\ hash\ compuation}{number\ of\ orginal\ procedure's\ hash\ compuation} = \frac{f}{f} = 1 \tag{7}$$

And the time complexity factor with the counters desynchronization similarly is

$$w_{cd} = \frac{f2^t}{f} = 2^t. \tag{8}$$

Although t is always small, w_{cd} is undesirable. Fortunately, the counters desynchronization case seldom happens, because this implies some attack or communication failure during the run of the reconnection procedure.

Both A and B in two mechanisms transmit and receive the same size of data (see Figures 5 and 10). Hence, we claim that the communication efficiency of the improved RPA mechanism is the same as that of the RPA mechanism. In storage cost, we merely consider the long-term storage overhead. Each device in the improved RPA mechanism requires extra $2t$ bits memory to realize two counters, compared with the RPA mechanism.

We know the energy cost of reconnection operation mainly depends on its time complexity, communication cost, and storage cost. According to the analysis above, we conclude that the energy overheads of our improved reconnection procedure have an insignificant increase compared with the RPA mechanism.

In addition, we simulate the reconnection procedures in both of the improved RPA mechanism and the RPA mechanism. The experimental platform is Windows 10 64 bits, Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz 3.19GHz, and 8.00 GB RAM. For Cryptographic tool, we use Python 3.6.6 cryptography toolkit PyCryptodemo. We set 20 records in the resolving list of each device. We also set t = 10. The experiment performs 1,000,000 runs of the reconnection procedure, records the run time, and averages them. Our experimental results are that the reconnection procedure of the RPA mechanism requires 0.031 ms, and the reconnection procedure of the improved RPA mechanism, respectively, requires 0.035 ms and 14.57 ms in the cases of synchronization and desynchronization. These time values nearly meet our theoretical results, i.e., Equations (7) and (8). To get more accuracy experimental results, these mechanisms should be implemented in an embedded Bluetooth platform. We leave this work in our future work.

5. Privacy Evaluation of Improved RPA Mechanism

Let $\{0, 1\}^*$ and $\{0, 1\}^L$ denote all finite binary strings and all binary strings with the L-bit length. Let Pr[] and Pr[|] be the probability and the conditional probability, respectively. Clearly, the privacy evaluation should focus on the reconnection procedures in RPA mechanisms. Similar to the ideas in [7,10,23], we use a formal privacy model to evaluate the improved RPA mechanism.

5.1. Model Definition

Let a set of Bluetooth LE devices be $I = \{1, 2, \ldots, s\}$. Π specifies how an initiator $i \in I$ and a non-initiator $j \in I$ behave during the reconnection run of an RPA mechanism. Let RPA_i denote i's RPA. For any i and j, let $\Pi_{i,j}$ denote i's instance of Π with j. $\Pi_{i,j}$ generates, transmits, and receives the packet(s) according to Π. $\Pi_{i,j}$ can be treated as an efficiently

computable function because it generates its RPA_i and resolves j's RPA_j. The internal state of $\Pi_{i,j}$ includes the following variables:

- *sid*: the unique identifier of $\Pi_{i,j}$;
- IRK_i and IRK_j: i's local IRK and i's peer IRK with j;
- IA_j: j's IA;
- LC_i and PC_i: i's local RPA counter and i's peer RPA counter with j;
- *tran*: a transcript of i's current run of $\Pi_{i,j}$ so far, i.e., the ordered set of packets transmitted and received by i so far;
- δ: a Boolean variable set to true or false denoting whether accepts or rejects at the end of the run of $\Pi_{i,j}$.

The run of Π can be modeled by collaboratively running $\Pi_{i,j}$ and $\Pi_{j,i}$. At the end of the run, i (resp. j) should either accept or reject the purported IA from j (resp. i), which is indicated by δ in $\Pi_{i,j}$ (resp. $\Pi_{j,i}$). Here, i (resp. j) needs to verify RPA_j (resp. RPA_i) by using IRK_j in $\Pi_{i,j}$ (resp. IRK_i in $\Pi_{j,i}$) and further link it to IA_j (resp. IA_i) in its resolving list. We can define the notion of correctness as follows.

Definition 1 (Correctness). *An RPA mechanism is correct if, given any honest initiator $i \in I$ and any honest non-initiator $j \in I$, the run of Π executed by the pair of $\Pi_{i,j}$ and $\Pi_{j,i}$ succeeds with overwhelming probability.*

Explanation. Correctness of a RPA mechanism means that if both $\Pi_{i,j}$ and $\Pi_{j,i}$ collaboratively generate *tran* using the same IRK_i and IRK_j, δs in both $\Pi_{i,j}$ and $\Pi_{j,i}$ are true at the end of the run. This notion is the customary requirement properly given honest participants. All RPA mechanisms must satisfy it. We can easily check that both the RPA mechanism in Section 2 and the improved RPA mechanism in Section 4 satisfy the correctness property.

5.1.1. Attacker

In order to model the actions of the attacker E, we assume that E can invoke a group of the oracles. That is, E sends the queries to the oracles and then receives the results from the oracles. With the help of the oracles, E is in complete control over all transmitted packets during the run of Π. For any $i, j \in I$, we define the following oracles.

Launch$(i, j) \rightarrow \{sid, \Pi_{i,j}, \Pi_{j,i}\}$: the Launch oracle initiates a run of Π, where *sid* is a unique identifier to identify the run. Both $\Pi_{i,j}$ and $\Pi_{j,i}$ maintain the same IRK_i and IRK_j. *tran*s in $\Pi_{i,j}$ and $\Pi_{j,i}$ are, respectively, set to empty value and δs in both $\Pi_{i,j}$ and $\Pi_{j,i}$ are set to false. In addition, if the RPA mechanism employs local RPA counter and peer RPA counter, LC_i, PC_i, LC_j, and PC_j are all set to 0. The Launch oracle actually simulates the initial connection procedure of the RPA mechanism.

Send$(m, sid, \Pi_{i,j}) \rightarrow m'$ (resp. Send$(m, sid, \Pi_{j,i}) \rightarrow m'$): the Send oracle advertises or transmits a packet m to i (resp. j) and receives an answer m' from j (resp. i). When m is valid according to $\Pi_{i,j}$ (resp. $\Pi_{j,i}$), m is also written into *tran* of $\Pi_{i,j}$ (resp. $\Pi_{j,i}$). Here, m, $m' \in \{1, 0\}^* \cup \{null\}$ and *null* denotes no transmitted packet. In addition, both LC_i and PC_i in $\Pi_{i,j}$ (resp. both LC_j and PC_j in $\Pi_{j,i}$) may be updated according to Π.

Execute$(i, j) \rightarrow \{\Pi_{i,j}, \Pi_{j,i}, tran, sid\}$: the Execute oracle automatically performs a complete run of Π between $\Pi_{i,j}$ and $\Pi_{j,i}$, where the run is identified by *sid* and *tran* stores all transmitted packets generated by the run.

Result$(\Pi_{i,j}) \rightarrow x$: the Result oracle outputs x to show whether $\Pi_{i,j}$ is successfully complete. That is, if $\Pi_{i,j}$'s δ is true, then $x = 1$; else $x = 0$.

Corrupt$(\Pi_{i,j},) \rightarrow \{i, RPA_i\}$ (resp. Corrupt$(\Pi_{j,i}) \rightarrow \{j, RPA_j\}$): If $\Pi_{i,j}$'s (resp. $\Pi_{j,i}$'s) δ is true, then the Corrupt oracle returns i (resp. j) and the corresponding RPA_i (resp. RPA_j) generated by $\Pi_{i,j}$ (resp. $\Pi_{j,i}$).

5.1.2. Privacy

Let k or 1^k be the security parameter. A real-valued function ε: $N \rightarrow [0, 1]$ is said negligible if for every polynomial pl there exists an integer N such that for every $k > N$, $\varepsilon(k) < 1/pl(k)$ holds.

As shown in Figure 12, we present the experiment $\text{Pri-Exp}_{\Pi, E}(k)$ to examine the privacy of Π. In the setup stage, a set of devices create AIs and generate the local random IRKs and obtain the peer IRKs and AIs. The devices set local RPA counters and peer RPA counters if required. At the end of the setup stage, all devices build their resolving lists to store the corresponding data. The training I stage allows the attacker E to invoke the Launch, Send, Execute, Result, and Corrupt oracles. Hence, E should learn how to run Π between any pair of the devices and collect a group of the device identities and the used RPAs. In the challenge stage, E needs to choose two devices j_0 and j_1 and submit them to the Test oracle. Upon receiving j_0 and j_1, the Test oracle flips a random coin bit $b \in \{0, 1\}$ and returns j_b back to E. The training II stage continuously allows E to invoke the Launch, Send, Execute, and Result oracles related to j_b. That is, E can manipulate j_b's runs of Π with any other device i. In the end, E is required to output his guessing bit of b. We define the privacy definition of the RPA mechanism using above experiment.

Definition 2 (Privacy). *Assume that* Π *is specified by an RPA mechanism. The RPA mechanism is private, if for any probabilistic polynomial time (PPT) attacker E,*

$$Pri\text{-}Adv_{\Pi, E} = |\, Pr[Pri\text{-}Exp_{\Pi, E}(k) = 1] - Pr[Pri\text{-}Exp_{\Pi, E}(k) = 0] \,| \quad (9)$$

is negligible in the security parameter k.

The experiment $\text{Pri-Exp}_{\Pi, E}(k)$

Stage 1. Setup

Create the resolving list for each $i \in I = \{1, 2, \ldots, s\}$ according to the initial connection procedure of Π.

(1) Generate and keep IA_i.

(2) For each of i's pairing $j \in I = \{1, 2, \ldots, s\}$ do

(2.1) Run random key generation $G(1^k) \rightarrow IRK_i$ and write IRK_i into the local IRK of its resolving list. Send IRK_i and IA_i to j.

(2.2) Receive IRK_j and IA_j from j and write IRK_j and IA_j into the peer IRK and IA of its resolving list.

(2.3) Set LC_i and RC_i and write them into the local RPA counter and the peer RPA counter of its resolving list if Π is required.

Stage 2. Training I

The attacker E can adaptively issue the oracles O = {Launch, Send, Execute, Result, Corrupt} to any pair of i and j, where $i, j \in I$.

Stage 3. Challenge

(1) E selects j_0 and j_1, where $j_0, j_1 \in I$.

(2) E calls an oracle Test(j_0, j_1).

(3) To answer the query, the oracle Test flips a random coin bit b and returns j_b to E.

Stage 4. Training II

E can adaptively issue the oracles O = {Launch, Send, Execute, Result} to interact with the pair of any i and j_b, where $i \in I$.

Stage 5. Guess

E outputs his guess bit b' for the random bit b.

Output. If $b = b'$, then the experiment $\text{Pri-Exp}_{\Pi, E}(k)$ outputs 1; else the experiment $\text{Pri-Exp}_{\Pi, E}(k)$ outputs 0.

Figure 12. Privacy experiment for RPA mechanisms.

Explanation. Definition 2 states that each E cannot detect which is the device (i.e., j_0 or j_1) was selected during the challenge stage with advantage significantly better than taking a random bit guess. An equivalent way of stating this definition is to that every E behaves the same way when it receives j_0 and when it receives j_1. Since E outputs a single bit, behaving the same way means that E outputs 1 with almost the same probability in each case.

We can apply the model proposed above to check and prove the privacy of RPA mechanisms. For example, we examine the RPA mechanism in Section 2. In the training I stage, E selects i and j and invokes Launch(i, j) to get their *sid*, $\Pi_{i,j}$, and $\Pi_{j,i}$ in current run. Then, E invokes the Corrupt oracle using $\Pi_{i,j}$ as the input and obtains i and the corresponding RPA_i. E submits $j_0 = i$ and $j_1 (\neq i, j) \in I$ and calls the oracle Test (j_0, j_1) in the challenge stage. During the training II stage, E calls the oracle Launch(j_b, j) to obtain *sid*, $\Pi_{jb,j}$, and $\Pi_{j,jb}$ in current run, and then invokes the oracle Send(RPA_i, *sid*, $\Pi_{j,jb}$) to wait j's response. E outputs the guess bit $b' = 0$, if j's response is received. Otherwise, he outputs the guess bit $b' = 1$. Let ν be the probability that j_1 also can generate the same RPA_i. We have

$$\begin{gathered}\text{Pri-Adv}_{\Pi,E} = |\Pr[\text{Pri-Exp}_{\Pi,E}(k) = 1] - \Pr[\text{Pri-Exp}_{\Pi,E}(k) = 0]| = \\ |\Pr[\text{Pri-Exp}_{\Pi,E}(k) = 1 \mid b = 0]\Pr[b = 0] + \Pr[\text{Pri-Exp}_{\Pi,E}(k) = 1 \mid b = 1]\Pr[b = 1] - \\ \Pr[\text{Pri-Exp}_{\Pi,E}(k) = 0 \mid b = 0]\Pr[b = 0] - \Pr[\text{Pri-Exp}_{\Pi,E}(k) = 0 \mid b = 1]\Pr[b = 1]| \leq \\ |1/2 + (1 - \nu)/2 - 0/2 - \nu/2| = 1 - v.\end{gathered} \tag{10}$$

According to the proposed model, the RPA mechanism fails to provide the privacy feature, because $1 - \nu$ is non-negligible.

5.2. Privacy Result of Improved RPA Mechanism and Its Proof

In the improved RPA mechanism, E can determine the identity of the targeted device and intercept the RPAs advertised or responded by the targeted device. Although E can replay these used RPAs during the reconnection procedure of the improved RPA mechanism, E cannot identify the targeted device by exploiting the counterpart's response. The reason is that the counterpart always fails to resolve the used RPAs due to its peer RPA counter, and E therefore cannot receive the counterpart's response. This privacy analysis is informal. We require the keyed pseudorandom function as the cryptographic tool to support the formal privacy analysis of the improved RPA mechanism. The security definition of the keyed pseudorandom function [31] is formally presented as follows.

Definition 3. *Let F: $\{0,1\}^k \times \{0,1\}^{l0} \rightarrow \{0,1\}^{l1}$ be an efficient keyed function. We say F is a pseudorandom function if for all PPT distinguishers D, there exists a negligible function ε such that*:

$$|\Pr[D(1^k, F(k,)) = 1] - \Pr[D(1^k, R()) = 1]| \leq \varepsilon(k), \tag{11}$$

where the k-bit key K is chosen uniformly at random, and R is chosen uniformly at keyed random from the set of random functions mapping l_0-bit strings to l_1-bit strings.

In the following, we have the privacy result of the improved RPA mechanism excluding the desynchronization of two devices.

Theorem 1. *Let Π be the reconnection procedure described as Figures 10 and 11 without considering exception process. Assume that each device keeps its resolving list secret. If the function ah is a keyed pseudorandom function as defined in Definition 3, Π is private according to Definition 2.*

Proof of Theorem 1. In the training II stage of Pri-Exp$_{\Pi,E}(k)$, E should interact with j_b. Let *Sim* be a simulator, which imitates j_b's behavior during the training II stage of Pri-Exp$_{\Pi,E}(k)$. However, *Sim* knows neither the random bit b nor j_b's secrets in its resolving list. We demonstrate that from view of E, *Sim* will be computationally indistinguishable from a real j_b. It states clearly that E cannot identify j_b at the guess stage, because E in Pri-Exp$_{\Pi,E}(k)$ gains no knowledge from its interaction with j_b.

In the challenge stage of Pri-Exp$_{\Pi,E}(k)$, we know that E chooses j_0 and j_1. Let LT and LT' denote the full *tran* list of both j_0 and j_1 in the training I stage and the full *tran* list of j_b in the training II stage. During the training II stage, *Sim* simulates the Launch, Send, Execute, and Result oracles to E as follows.

Launch oracle. Consider E calls the Launch oracle using i and j_b to *Sim*. *Sim* invokes Launch(i, j_b) to get *sid*, $\Pi_{i,jb}$, $\Pi_{jb,i}$. Then, *Sim* further sends *sid*, $\Pi_{i,fjb}$, $\Pi_{fjb,i}$ to E, where $\Pi_{i,fjb}$ (resp. $\Pi_{fjb,i}$) are *Sim*'s simulating function of $\Pi_{i,jb}$ (resp. $\Pi_{jb,i}$).

Send oracle. (1) j_b is an initiator. (1.1) When E invokes Send(*null*, *sid*, $\Pi_{fjb,i}$), *Sim* generates a 24-bit *prand*′ as Figure 6 and a random 24-bit *hash*′. *Sim* forms RPA'_{jb}= *hash*′||*prand*′ and sends it to E, and then further records RPA'_{jb} in its LT'. In addition, *Sim* calls Send(*null*, *sid*, $\Pi_{jb,i}$) to obtain RPA_{jb}. (1.2) When E invokes Send(RPA'_{jb}, *sid*, $\Pi_{i,fjb}$), *Sim* calls Send(RPA_{jb}, *sid*, $\Pi_{i,jb}$) and returns the output of Send(RPA_{jb}, *sid*, $\Pi_{i,jb}$), i.e., RPA_i, to E. Then, *Sim* records RPA_i in its LT'. (1.3) When E transmits other RPA by the Send oracle, *Sim* simply calls i's Send oracle with the receiving RPA and returns i's output back to E. (2) j_b is a non-initiator. (2.1) When E invokes Send(*null*, *sid*, $\Pi_{i,fjb}$), *Sim* calls Send(*null*, *sid*, $\Pi_{i,jb}$) to get RPA_i and returns it to E. Then, *Sim* records RPA_i in its LT'. (2.2) When E invokes Send(RPA_i, *sid*, $\Pi_{fjb,i}$), *Sim* generates a 24-bit *prand*′ as Figure 6 and a random 24-bit *hash*′. Then, *Sim* forms RPA'_{jb} = *hash*′||*prand*′ and records RPA'_{jb} in its LT, and then further sends it to E. (2.3) When E transmits other RPA by the Send oracle, *Sim* has no response.

Execute oracle. When E calls the Execute oracle to *Sim*, *Sim* calls Execute (i, j_b) to generate *sid*, RPA_i, and RPA_{jb}. Then, *Sim* generates a 24-bit *prand*′ and a random 24-bit *hash*′ and forms RPA'_{jb} = *hash*′||*prand*′. *Sim* records {*sid*, RPA_i, RPA'_{jb}} in its LT', and then sends them to E.

Result oracle. When E calls Result($\Pi_{i,fjb}$) (resp. Result($\Pi_{fjb,i}$)), *Sim* returns 1 to E if all input and output of $\Pi_{i,fjb}$ (resp. $\Pi_{fjb,i}$) are in its LT'. Otherwise, *Sim* returns 0 to E.

To distinguish *Sim*'s training II stage and a real training II stage, E must determine that at least a run is invalid for j_b and i. We know that both j_b and i respectively maintain the local RPA counters LC_{jb} and LC_i and the peer RPA counters RC_{jb} and RC_i. Hence, E cannot invoke the Corrupt oracle and find out *Sim* by reusing j_b's RPAs in the training I stage. If E determines that *Sim* exists, he must rule out at least one {*sid*, RPA_i, RPA'_{jb}} in the training II stage. Let $q(k)$ denote at most queries of the Send oracle, the Execute oracle, and the Corrupt oracle during each training stage of the experiment Pri-Exp$_{\Pi,E}(k)$. $q(k)$ should be a polynomial function. To figure *Sim* out, one of the following two cases must occur at some point when E plays Pri-Exp$_{\Pi,E}(k)$.

Case 1. Consider two RPAs $RPA_{jb} \in LT$ and $RPA'_{jb} \in LT'$, where RPA_{jb}=*hash*||*prand*, RPA'_{jb} = *hash*′||*prand*′, and *hash*, *prand*, *hash*′, and *prand*′ are 24 bits. According to Equation (4), we know *hash* = *ah*(IRK_{jb}, *padding*1||LC_{jb}||*prand*) mod 2^{24}, where IRK_{jb} is j_b's IRK and LC_{jb} is j_b's local RPA counter. However, *Sim* randomly generates *hash*′ in RPA'_{jb}. Hence, when *prand* = *prand*′, E finds out *Sim* by verifying *hash* = *hash*′. Let $|LT|$ and $|LT'|$ denote respectively the numbers of RPA_{jb} and RPA'_{jb}. We have $|LT| \leq q(k)$ and $|LT'| \leq q(k)$, because E makes at most $q(k)$ Send and Execute calls in each corresponding training stage. In the improved RPA mechanism, we know RPA_{jb} and RPA'_{jb} are all 48-bit values, and *prand* and *prand*′ have two constant bits. We have that the RPA space is 2^{46}. It thus follows that this condition occurs with the probability at most $q(k)^2/2^{46}$.

Case 2. *Sim* randomly chooses *hash*′ to form j_b's RPA during the training II stage. Comparatively, j_b should use the function *ah* to generate *hash*. Hence, in order to recognize *Sim*, E can use LT and LT' to distinguish this distribution difference. Let Ev^* be the event that E succeeds in distinguishing *Sim* and j_b according to their RPAs in LT and LT'. We know that *ah* is a keyed pseudorandom function as Definition 3. Therefore, we can directly construct the distinguisher D_1 to tell the keyed pseudorandom function *ah* from the truly random function R. According to Equation (11), the probability that E can figure out *Sim* by the pseudorandom characteristic is

$$\begin{aligned} \Pr[Ev^*] \leq\ & |\Pr[D_1(1^k, ah(k,)) = 1] - \Pr[D_1(1^k, R()) = 1]| = \\ & |\Pr[D(1^k, F(k,)) = 1] - \Pr[D(1^k, R()) = 1]| \leq \varepsilon(k). \end{aligned} \tag{12}$$

Hence, the polynomially bounded E can distinguish Sim from real j_b with the negligible probability at most $q(k)^2/2^{46} + \varepsilon(k)$. □

6. Conclusions

The Bluetooth standard specifies the RPA mechanism to protect the address privacy of LE devices because a growing number of people use them for the sensitive transactions. However, we reported the traceability weakness of the RPA mechanism. We therefore proposed an improved RPA mechanism and proved it correctly repairs the traceability weakness existing in the RPA mechanism. The improved RPA mechanism is easy to implement in LE devices, though with a little extra overhead. We believe that our research result is a steady step to enhancing the address privacy of LE devices.

In this work, we do not implement the improved RPA mechanism in Bluetooth simulation platform. Hence, one future work is to use software-defined radios to realize the RPA mechanism and our improvement. We hope to obtain more accurate energy costs for both mechanisms. We do not investigate the forward/backward privacy of the RPA mechanism, which also is our future work. To solve this problem, we need to develop a new formal privacy model and redesign the RPA mechanism.

Author Contributions: Conceptualization, D.S.; methodology, D.S. and Y.T.; validation, D.S. and Y.T.; formal analysis, D.S.; investigation, D.S.; writing—original draft preparation, D.S. and Y.T.; writing—review and editing, D.S. and Y.T.; supervision, D.S.; funding acquisition, D.S. All authors have read and agreed to the published version of the manuscript.

Funding: The work of Dazhi Sun was supported in part by the National Natural Science Foundation of China under Grant No. 61872264. The APC was funded by the National Natural Science Foundation of China under Grant No. 61872264.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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

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

References

1. *IEEE Std 802.15.1*; IEEE Standard for Telecommunications and Information Exchange between Systems-LAN/MAN-Specific Requirements-Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs). IEEE: Piscataway, NJ, USA, 2002. Available online: https://ieeexplore.ieee.org/document/1016473 (accessed on 10 October 2022).
2. Talasila, M.; Curtmola, R.; Borcea, C. Collaborative Bluetooth-based location authentication on smart phones. *Pervasive Mob. Comput.* **2015**, *17*, 43–62. [CrossRef]
3. Draghici, A.; Van Steen, M. A survey of techniques for automatically sensing the behavior of a crowd. *ACM Comput. Surv.* **2018**, *51*, 21. [CrossRef]
4. Specification of the Bluetooth System, Covered Core Package Version: 4.0, Master Table of Contents & Compliance Requirements, Bluetooth SIG Proprietary. Available online: https://www.bluetooth.com/specifications/specs/core-specification-4-0 (accessed on 10 October 2022).
5. Specification of the Bluetooth System, Covered Core Package Version: 5.2, Master Table of Contents & Compliance Requirements, Bluetooth SIG Proprietary. Available online: https://www.bluetooth.com/specifications/specs/core-specification-5-2 (accessed on 10 October 2022).
6. Specification of the Bluetooth System, Covered Core Package Version: 5.3, Master Table of Contents & Compliance Requirements, Bluetooth SIG Proprietary. Available online: https://www.bluetooth.com/specifications/specs/core-specification-5-3 (accessed on 10 October 2022).

7. 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 18th International Conference on Information and Communications Security (ICICS'16), Singapore, 29 November–2 December 2016; Lam, K.Y., Chi, C.H., Eds.; Lecture Notes in Computer Science. Springer: Cham, Switzerland, 2016; Volume 9977, pp. 403–417.
8. 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]
9. Hassan, S.S.; Bibon, S.D.; Hossain, M.S.; Atiquzzaman, M. Security threats in Bluetooth technology. *Comput. Secur.* **2018**, *74*, 308–322. [CrossRef]
10. 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]
11. Padgette, J.; Bahr, J.; Batra, M.; Holtmann, M.; Smithbey, R.; Chen, L.; Scarfone, K. Guide to Bluetooth Security. National Institute of Standards and Technology, U.S. Department of Commerce, Special Publication 800-121 Revision 2, May 2017. Available online: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf (accessed on 10 October 2022).
12. Cȁsar, M.; Pawelke, T.; Steffan, J.; Terhorst, G. A survey on Bluetooth low energy security and privacy. *Comput. Netw.* **2022**, *205*, 108712. [CrossRef]
13. Gibbs, J. BLE and Laird's BL6x0 Series & BT900 Modules: A Guide to Security and Privacy. EECatalog. 2014. Available online: https://www.lairdconnect.com/resources/white-papers/ble-and-lairds-bl6x0-series-bt900-modules-guide-security-and-privacy (accessed on 10 October 2022).
14. AN99209. PSoC®4 BLE and PRoC™ BLE: Bluetooth LE 4.2 Features. CYPRESS. 2017. Available online: https://www.cypress.com/file/224826/download (accessed on 10 October 2022).
15. Wang, P. Bluetooth Low Energy—Privacy Enhancement for Advertisement. Ph.D. Thesis, Norwegian University of Science and Technology, Trondheim, Norway, June 2014.
16. 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'16), St. Augustine, FL, USA, 23–24 February 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 99–104.
17. 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; USENIX Association: Berkeley, CA, USA, 2016; pp. 1205–1221.
18. Issoufaly, T.; Tournoux, P.U. BLEB: Bluetooth Low Energy Botnet for Large Scale Individual Tracking. In Proceedings of the 1st International Conference on Next Generation Computing Applications (NextComp'17), Flic-en-Flac, Mauritius, 19–21 July 2017; IEEE: New York, NY, USA, 2017; pp. 115–120.
19. 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; Association for Computing Machinery: New York, NY, USA, 2018; pp. 43–52.
20. Wu, J.L.; Nan, Y.H.; Kumar, V.; Payer, M.; Xu, D.Y. BlueShield: Detecting Spoofing Attacks in Bluetooth Low Energy Networks. In Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID'20), San Sebastian, Spain, 14–16 October 2020; USENIX Association: Berkeley, CA, USA, 2020; pp. 397–411.
21. Ludant, N.; Vo-Huu, T.D.; Narain, S.; Noubir, G. Linking Bluetooth LE & Classic and Implications for Privacy-Preserving Bluetooth-Based Protocols. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP'21), Virtual Event, 24–27 May 2021; IEEE Computer Society: Los Alamitos, CA, USA, 2021; pp. 1318–1331.
22. Zhang, Y.; Lin, Z.Q. When Good Becomes Evil: Tracking Bluetooth Low Energy Devices via Allowlist-Based Side Channel and Its Countermeasure. In Proceedings of the 29th ACM Conference on Computer and Communications Security (CCS'22), Los Angeles, CA, USA, 7–11 November 2022; Association for Computing Machinery: New York, NY, USA, 2022; pp. 3181–3194.
23. Sun, D.Z.; Sun, L.; Yang, Y. On secure simple pairing in Bluetooth standard v5.0-part ii: Privacy analysis and enhancement for low energy. *Sensors* **2019**, *19*, 3259. [CrossRef] [PubMed]
24. Zhang, Y.; Weng, J.; Dey, R.; Jin, Y.E.; Lin, Z.Q.; Fu, X.W. Breaking Secure Pairing of Bluetooth Low Energy Using Downgrade Attacks. In Proceedings of the 29th USENIX Security Symposium (USENIX Security'20), Virtual Event, 12–14 August 2020; USENIX Association: Berkeley, CA, USA, 2020; pp. 37–54.
25. von Tschirschnitz, M.; Peuckert, L.; Franzen, F.; Grossklags, J. Method Confusion Attack on Bluetooth Pairing. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP'21), Virtual Event, 24–27 May 2021; IEEE Computer Society: Los Alamitos, CA, USA, 2021; pp. 1332–1347.
26. Bello-Ogunu, E.; Shehab, M.; Miazi, N.S. Privacy Is the Best Policy: A Framework for BLE Beacon Privacy Management. In Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC'19), Milwaukee, WI, USA, 15–19 July 2019; IEEE Computer Society: New York, NY, USA, 2019; pp. 823–832.
27. Chen, Z.; Hu, H.B.; Yu, J.L. Privacy-Preserving Large-Scale Location Monitoring Using Bluetooth Low Energy. In Proceedings of the 11th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN'15), Shenzhen, China, 16–18 December 2015; IEEE Computer Society: New York, NY, USA, 2015; pp. 69–78.
28. 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]

29. *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)*; IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture. IEEE: Piscataway, NJ, USA, 2014. Available online: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6847097 (accessed on 10 October 2022).
30. *FIPS 197*; Specification for the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication (FIPS PUB) 197. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001. Available online: https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf (accessed on 10 October 2022).
31. Katz, J.; Lindell, Y. *Introduction to Modern Cryptography: Principles and Protocols*, 3rd ed.; Chapman & Hall/CRC: Boca Raton, FL, USA, 2020; pp. 75–84.