Rethinking the Security and Privacy of Bluetooth Low Energy

Zhiqiang Lin
Distinguished Professor of Engineering
zlin@cse.ohio-state.edu

11/28/2022
Outline

1. Introduction
2. BLE Security
3. BLE Privacy
4. Takeaway
Outline

1 Introduction

2 BLE Security

3 BLE Privacy

4 Takeaway
What is Bluetooth
What is Bluetooth

Total Annual Bluetooth® Device Shipments

Numbers in billions

2017: 3.6
2018: 3.8
2019: 4.1
2020: 4.1
2021: 4.7
2022: 5.1
2023: 5.5
2024: 6.0
2025: 6.5
2026: 7.0

9% CAGR

Data Source: ABI Research, 2022
Why Named Bluetooth

**Harald “Bluetooth” Gormsson**

- King of Denmark 940-981.
- He was also known for his bad **tooth**, which had a very dark **blue-grey** shade.
- He united the Tribes of Denmark.

The Bluetooth wireless specification design was named after the king in 1997, based on an analogy that the technology would unite devices the way Harald Bluetooth united the tribes of Denmark into a single kingdom.
History of Bluetooth

1994
Bluetooth Prototype
Bluetooth SIG
 (~700KB/s)

1998
Bluetooth 1.0+1.0b
 (~700KB/s)

1999
Bluetooth 1.1
 (~700KB/s)

2001
Bluetooth 2.0/2.1 +EDR
 (~2.1MB/s)

2004
Bluetooth 3.0+HS
 (~24MB/s)

2009
Bluetooth 4.0
 (~24MB/s)

2010
Bluetooth 4.1
 (~50MB/s)

2013
Bluetooth 4.2
 (~50MB/s)

2014
Bluetooth 5.0
 (~50MB/s)

2016
Bluetooth 5.1/5.2
 (~50MB/s)

2019
Our Recent Works on Bluetooth Security and Privacy

Bluetooth Prototype
1994
Bluetooth SIG
1998
1999
Bluetooth 1.0+1.0b
Bluetooth 1.1
2001
(~700KB/s)
2004
2009
2010
2013
2014
2016
Bluetooth 2.0/2.1 +EDR
Bluetooth 3.0+HS
Bluetooth 4.0
Bluetooth 4.1
Bluetooth 4.2
Bluetooth 5.0
Bluetooth 5.1/5.2
2019
(~24MB/s)
(~24MB/s)
(~50MB/s)
(~50MB/s)
(~2.1MB/s)
(~50MB/s)
Our Recent Works on Bluetooth Security and Privacy

1. **BLEScope**: Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps. In ACM CCS 2019
2. **FirmXRay**: Detecting Bluetooth Link Layer Vulnerabilities From Bare-Metal Firmware. In ACM CCS 2020.
4. **When Good Becomes Evil: Tracking Bluetooth Low Energy Devices via Allowlist-based Side Channel and Its Countermeasure**. In ACM CCS 2022 (Best paper award honorable mention)
5. **Extrapolating Formal Analysis to Uncover Attacks in Bluetooth Passkey Entry Pairing**. In NDSS 2023
Our Recent Works on Bluetooth Security and Privacy

1. BLEScope: Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps. In ACM CCS 2019


4. When Good Becomes Evil: Tracking Bluetooth Low Energy Devices via Allowlist-based Side Channel and Its Countermeasure”. In ACM CCS 2022 (Best paper award honorable mention)

5. Extrapolating Formal Analysis to Uncover Attacks in Bluetooth Passkey Entry Pairing. In NDSS 2023
1 Introduction
2 BLE Security
3 BLE Privacy
4 Takeaway
Pairing Workflow

Device

OS

App
Pairing Workflow

1. Start pairing

Device → OS

OS → App
Pairing Workflow

I/O Features
- Keypad
- Screen
- Out of band Channel
Pairing Workflow

1. Start pairing

2. Pairing feature exchange

3. Authentication and encryption

Pairing Methods
- Just Works
- Passkey Entry
- Out of band
- Numeric Comparison
Pairing Workflow

Pairing Methods
- Just Works
- Passkey Entry
- Out of band
- Numeric Comparison
Pairing Workflow

1. Start pairing
2. Pairing feature exchange
3. Authentication and encryption
4. Key distribution (e.g. IRK)
5. Encrypted communication

Pairing Methods
- Just Works
- Passkey Entry
- Out of band
- Numeric Comparison
Workflow of Pairing: Elliptic Curve Diffie–Hellman (ECDH) Key Exchange

1. Alice generates a random ECC key pair: $\{Pri_A, PK_A = Pri_A \times G\}$
Workflow of Pairing: Elliptic Curve Diffie–Hellman (ECDH) Key Exchange

1. Alice generates a random ECC key pair: \( \{Pri_A, PK_A = Pri_A \times G\} \)
2. Bob generates a random ECC key pair: \( \{Pri_B, PK_B = Pri_B \times G\} \)
Workflow of Pairing: Elliptic Curve Diffie–Hellman (ECDH) Key Exchange

1. Alice generates a random ECC key pair: \( \{Pri_A, PK_A = Pri_A \cdot G\} \)
2. Bob generates a random ECC key pair: \( \{Pri_B, PK_B = Pri_B \cdot G\} \)
3. Alice and Bob exchanges \( PK_A \) and \( PK_B \)
Workflow of Pairing: Elliptic Curve Diffie–Hellman (ECDH) Key Exchange

1. Alice generates a random ECC key pair: \( \{Pri_A, PK_A = Pri_A \times G\} \)
2. Bob generates a random ECC key pair: \( \{Pri_B, PK_B = Pri_B \times G\} \)
3. Alice and Bob exchanges \( PK_A \) and \( PK_B \)
4. Alice calculates sharedKey: \( K_A = Pri_A \times PK_B \)
Workflow of Pairing: Elliptic Curve Diffie–Hellman (ECDH) Key Exchange

1. Alice generates a random ECC key pair: \( \{Pri_A, PK_A = Pri_A \times G\} \)
2. Bob generates a random ECC key pair: \( \{Pri_B, PK_B = Pri_B \times G\} \)
3. Alice and Bob exchanges \( PK_A \) and \( PK_B \)
4. Alice calculates sharedKey: \( K_A = Pri_A \times PK_B \)
5. Bob calculates sharedKey: \( K_B = Pri_B \times PK_A \)
Workflow of Pairing: Elliptic Curve Diffie–Hellman (ECDH) Key Exchange

1. Alice generates a random ECC key pair: \( \{\text{Pri}_A, \text{PK}_A = \text{Pri}_A \times G\} \)
2. Bob generates a random ECC key pair: \( \{\text{Pri}_B, \text{PK}_B = \text{Pri}_B \times G\} \)
3. Alice and Bob exchanges \( \text{PK}_A \) and \( \text{PK}_B \)
4. Alice calculates sharedKey: \( K_A = \text{Pri}_A \times \text{PK}_B \)
5. Bob calculates sharedKey: \( K_B = \text{Pri}_B \times \text{PK}_A \)

\[ \text{Pri}_A \times (\text{Pri}_B \times G) = \text{Pri}_B \times (\text{Pri}_A \times G) \]
Workflow of Passkey Entry
Workflow of Passkey Entry
Workflow of Passkey Entry

Device A

123456

PK_A

Device B

PK_B

PK_A

PK_B
Workflow of Passkey Entry

Device A

123456

Device B

PK_A

PK_B

123456
Workflow of Passkey Entry

Device A

123456

PK_A

Device B

123456

PK_B

F(Pri_A, PK_B, 123456) F(Pri_A, PK_B, 123456)
Workflow of Passkey Entry

Device A

123456

PK_A

PK_M

Device B

PK_B

PK_M

123456

PK_A=F(Pri_A,PK_M,123456)  
PK_B=F(Pri_B,PK_M,123456)

K_A=F(Pri_A,PK_M,123456)

K_B=F(Pri_B,PK_M,123456)
Workflow of Numeric Comparison
Workflow of Numeric Comparison

Device A

PK_A

PK_B

Device B

PK_B

PK_A

Workflow:
1. Device A sends its public key PK_A to Device B.
2. Device B sends its public key PK_B to Device A.
3. Both devices compare the PKs. If they match, the connection is secure.
Workflow of Numeric Comparison

Device A

Device B

PK_A

PK_B

Hash(PK_A, PK_B)

Hash(PK_A, PK_B)

123456

123456
Workflow of Numeric Comparison

Device A

PK_A

PK_B

Hash(PK_A, PK_B)

123456

Device B

PK_A

PK_B

Hash(PK_A, PK_B)

123456
Workflow of Numeric Comparison

Device A

Device B

PK_A

PK_B

Hash(PK_A, PK_B)

123456

F(Pri_A, PK_B, 123456)

Hash(PK_A, PK_B)

123456

F(PK_A, Pri_B, 123456)
Workflow of Numeric Comparison

\[
K_A = F(Pri_A, PK_M, 123456)
\]

\[
K_B = F(Pri_B, PK_M, 123456)
\]
Workflow of Out of Band

Device A

Device B
Workflow of Out of Band

Device A

PK_A

Device B
Workflow of Out of Band

Device A

PK_A

PK_B

Device B
Workflow of Out of Band
Workflow of Out of Band

\[
K = F(Pri_A, PK_B, S)
\]

\[
K = F(Pri_B, PK_A, S)
\]
Workflow of Out of Band

\[ K_A = F(Pri_A, PK_M, S) \]
\[ K_B = F(Pri_B, PK_M, S) \]
Workflow of Justworks
Workflow of Justworks
Workflow of Justworks

Device A

Device B

PK_A

PK_B
Workflow of Justworks

\[ K = F(Pri_A, PK_M, 00000) \]

\[ K = F(Pri_B, PK_M, 000000) \]
Workflow of Justworks

Device A

PK_A

PK_M

PK_B

F(PK_A, Pri_M, 00000)

K_A = F(Pri_A, PK_M, 00000)

Device B

PK_M

PK_B

F(PK_B, Pri_M, 00000)

K_B = F(Pri_B, PK_M, 00000)

Malicious Device
Our Downgrade Attacks against Bluetooth Low Energy

1. Start authentication
2. Security feature exchange
3. 2G Authentication
4. Disconnect
4. Communication
Our Downgrade Attacks against Bluetooth Low Energy

1. Start pairing
2. Pairing feature exchange
3. Weak Authentication
   - Just Works
4. Disconnect
4. Communication
Our Downgrade Attacks against Bluetooth Low Energy

Paired with a secure pairing method (Passkey Entry/Numeric Comparison)
Our Downgrade Attacks against Bluetooth Low Energy

1. Impersonate the victim device and deploy attacks against the mobile

Paired with a secure pairing method (Passkey Entry/Numeric Comparison)
Our Downgrade Attacks against Bluetooth Low Energy

1. Impersonate the victim device and deploy attacks against the mobile
2. Use the stolen information (i.e., IRK) to create a Fake mobile

Paired with a secure pairing method (Passkey Entry/Numeric Comparison)
Our Downgrade Attacks against Bluetooth Low Energy

1. Impersonate the victim device and deploy attacks against the mobile

2. Use the stolen information (i.e., IRK) to create a Fake mobile

Paired with a secure pairing method (Passkey Entry/Numeric Comparison)
Our Downgrade Attacks against Bluetooth Low Energy

1. Impersonate the victim device and deploy attacks against the mobile.
2. Use the stolen information (i.e., IRK) to create a fake mobile.
3. Deploy attacks against the device.

Paired with a secure pairing method (Passkey Entry/Numeric Comparison)
Our Downgrade Attacks against Bluetooth Low Energy

The Tested BLE devices

MITM attack against BLE keyboards

CVE-2020-9770
Our First Finding: Allowlist-based Side Channel
Our First Finding: Allowlist-based Side Channel
Our First Finding: Allowlist-based Side Channel
Our First Finding: Allowlist-based Side Channel
Our First Finding: Allowlist-based Side Channel
Our First Finding: Allowlist-based Side Channel
Our First Finding: Allowlist-based Side Channel
Our First Finding: Allowlist-based Side Channel
Our Second Finding: MAC Address Can be Replayed

Pairing (Exchange Identity Resolving Key)
Our Second Finding: MAC Address Can be Replayed

Pairing (Exchange Identity Resolving Key)

Random Address (RA) Generation

\[ RA_p = pr_{rand24} || H_{24}(pr_{rand24} || IRK_p) \]

Random Address (RA) Resolution

\[ 47:2B:3C:6F:1C:DE \]
Our Second Finding: MAC Address Can be Replayed

Pairing (Exchange Identity Resolving Key)

Random Address (RA) Generation

\[ RA_p = \text{prand}_{24} || H_{24}(\text{prand}_{24} || IRK_p) \]

Random Address (RA) Resolution

\[ RA_c = \text{prand}_{24} || H_{24}(\text{prand}_{24} || IRK_c) \]

Verification Results

\[ RA_p \]
Our Second Finding: MAC Address Can be Replayed

Pairing (Exchange Identity Resolving Key)

Random Address (RA) Generation

$RA_p = \text{prand}_{24} || H_{24}(\text{prand}_{24} || IRK_p)$

Random Address (RA) Resolution

$RA_c = \text{prand}_{24} || H_{24}(\text{prand}_{24} || IRK_c)$

Random Address (RA) Replay

$RA'_p = RA_p$

Verification Results

$RA_p$  

$47:2B:3C:6F:1C:DE$
Our Second Finding: MAC Address Can be Replayed

Random Address (RA) Generation

$$RA_p = \text{prand}_{24} || H_{24}(\text{prand}_{24} || IRK_p)$$

Random Address (RA) Resolution

$$RA_c = \text{prand}_{24} || H_{24}(\text{prand}_{24} || IRK_c)$$

Verification Results

$$RA_p = RA_p$$

$$RA_p' = RA_p$$

$$RA_c = RA_c$$
Attack Example

Tracking a Victim’s Real-time Location
Attack Example

Tracking a Victim’s Real-time Location
Attack Example

Tracking a Victim’s Real-time Location
Attack Example

Tracking a Victim’s Real-time Location
Attack Example

Tracking a Victim’s Real-time Location
Attack Example

Tracking a Victim’s Real-time Location
Attack Example

Tracking a Victim’s Real-time Location
Devices That are Subject to BAT Attacks

<table>
<thead>
<tr>
<th>Brand &amp; Model</th>
<th>Type &amp; OS</th>
<th>MAC Addr</th>
<th>Random Interval</th>
<th>Passive Attacks</th>
<th>Active Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Pixel 4</td>
<td>Phone (Android 11)</td>
<td>RPA</td>
<td>5-15</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>Google Pixel 2</td>
<td>Phone (Android 10)</td>
<td>RPA</td>
<td>5-15</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>Samsung S10</td>
<td>Phone (Android 10)</td>
<td>RPA</td>
<td>5-15</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>Google Pixel 4</td>
<td>Phone (Android 10)</td>
<td>RPA</td>
<td>5-15</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>iPhone 8</td>
<td>Phone (iOS 13.2)</td>
<td>RPA</td>
<td>15</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>iPhone 11</td>
<td>Phone (iOS 13.2)</td>
<td>RPA</td>
<td>15</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>iPad</td>
<td>Tablet (iOS 13.2)</td>
<td>RPA</td>
<td>15</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>Dell G2044KU</td>
<td>Laptop (Windows 10)</td>
<td>PA</td>
<td>-</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>Dell</td>
<td>Laptop (Ubuntu 20.04)</td>
<td>PA</td>
<td>-</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>Thinkpad T450s</td>
<td>Laptop (Windows 8)</td>
<td>PA</td>
<td>-</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
<tr>
<td>Surface Pro</td>
<td>Tablet (Windows 10)</td>
<td>PA</td>
<td>-</td>
<td>✓ ✓ ✓ ✓ ✓ ✓</td>
<td></td>
</tr>
</tbody>
</table>
Responsible Disclosure
The Bluetooth SIG has reserved CVE-2020-35473 for tracking this vulnerability post-publication. The Bluetooth SIG is beginning rollout of our recommendations on this privacy vulnerability report to the broader Bluetooth SIG membership.
The Bluetooth SIG has reserved CVE-2020-35473 for tracking this vulnerability post-publication. The Bluetooth SIG is beginning rollout of our recommendations on this privacy vulnerability report to the broader Bluetooth SIG membership.

“I wanted to confirm we are tracking the BT SIG recommendations for handling the Whitelisting and Resolvable MAC Address Randomization vulnerability.”
"The Bluetooth SIG has reserved CVE-2020-35473 for tracking this vulnerability post-publication. The Bluetooth SIG is beginning rollout of our recommendations on this privacy vulnerability report to the broader Bluetooth SIG membership."

"I wanted to confirm we are tracking the BT SIG recommendations for handling the Whitelisting and Resolvable MAC Address Randomization vulnerability."

"The Android security team has conducted an initial severity assessment on this report. Based on our published severity assessment matrix (1) it was rated as High severity. This issue has been assigned to the appropriate team for remediation, and we’re targeting a fix for release in an upcoming Android Security Bulletin."
"The Bluetooth SIG has reserved CVE-2020-35473 for tracking this vulnerability post-publication. The Bluetooth SIG is beginning rollout of our recommendations on this privacy vulnerability report to the broader Bluetooth SIG membership."

"I wanted to confirm we are tracking the BT SIG recommendations for handling the Whitelisting and Resolvable MAC Address Randomization vulnerability."

"The Android security team has conducted an initial severity assessment on this report. Based on our published severity assessment matrix (1) it was rated as High severity. This issue has been assigned to the appropriate team for remediation, and we’re targeting a fix for release in an upcoming Android Security Bulletin."
Our Countermeasure: Securing Address of BLE (SABLE)

Allowlist Side Channel (Mitigation)
- We advocate the use of an interval unpredictable, central and peripheral synchronized RPA generation scheme to mitigate the side channel.

MAC Address Replay (Prevention)
- We propose adding a sequence number (which could be a timestamp) when generating the RPA to ensure that each MAC address can only be used once to prevent the replay attack.
Performance of SABLE

Lesson Learned (1/3): BLE Communication Can Be Downgraded

- Bluetooth low energy (BLE) pairing can be **downgraded**
- There are many stages that are not part of the pairing process, but they are, in fact, closely related to pairing security.
- A systematic analysis of the pairing process, including the **error handling** of BLE communication, is needed.
Lesson Learned (2/3): New Features Need Re-examinations

- Public Address
- Resolvable Random
- Private Address (RPA)
- Allowlist
  - (Public Address)
  - (RPA)

- Bluetooth 1.0
- 2002
- 2010.6
- 2014.12
- Bluetooth 4.0
- Allowlist
  - (RPA)
Lesson Learned (2/3): New Features Need Re-examinations

BLE introduces multiple new features, some of which may violate existing assumptions.

Similar to allowlist, those new features need to be scrutinized. For example, Cross-transport key derivation (CTKD); Authorization; The Connection Signature Resolving Key (CSRK).
The specification (3,000+ pages) is often confusing and inconsistent across chapters.

The confusion may lead to different vendors implement BLE protocols in quite different ways, for example, for error handling, and IRK use.

Converting the Bluetooth specification to formal model (e.g., using NLP), and formally verify the entire protocol would help.

See our NDSS’23 paper.
Our Recent Work on Bluetooth Security and Privacy

1. BLEScope: Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps. In ACM CCS 2019
4. When Good Becomes Evil: Tracking Bluetooth Low Energy Devices via Allowlist-based Side Channel and Its Countermeasure”. In ACM CCS 2022 (Best paper award honorable mention)
5. Extrapolating Formal Analysis to Uncover Attacks in Bluetooth Passkey Entry Pairing. In NDSS 2023
Rethinking the Security and Privacy of Bluetooth Low Energy

Zhiqiang Lin
Distinguished Professor of Engineering
zlin@cse.ohio-state.edu

11/28/2022